Feature Article

SunScreen: Visual Fault Detection for Solar-Thermal Systems

Lukas Feierl, *SOLID Solar Energy Systems GmbH., 8020, Graz, Austria*

Torsten Möller, *University of Vienna, 1090, Vienna, Austria*

Peter Luidolt, *SOLID Solar Energy Systems GmbH., 8020, Graz, Austria*

Abstract—Fault detection is essential to ensure the proper operation of solar-thermal plants. Hence, monitoring personnel frequently analyze the data to detect unusual behavior. While visualization approaches may considerably support the monitoring personnel during their work, no existing application can yet deal with the multivariate and time-dependent sensor data, or does not fully support the users' workflow. Thus, this work introduces the visual framework SunScreen. It allows users to explore the sensor data, automatically detected anomalies, and system events (e.g., already detected faults and services). The feedback from the users shows that they appreciate the tool and especially its annotation functionality. However, the SUS results indicate that it does not meet all requirements yet.

Solar-thermal plants use solar irradiation to produce heat. With heat accounting for approximately 50% of the global energy demand,[1] this technology could play a crucial role in the transition to renewable energy. However, monitoring is essential to ensure the proper operation of the plants. Thus, monitoring personnel frequently analyze the sensor data to detect unusual behavior.

However, the main challenge for this fault detection is the complex behavior of solar-thermal plants and its multidimensional, time-dependent, and non-linear measurement data. As a result, the data is hard to interpret even by experts, while automatic fault detection algorithms are prone to raise false alarms.[2,3] Visual frameworks may considerably support the monitoring personnel by combining domain knowledge and automatic anomaly detection and letting users explore the measurement data.

Unfortunately, no designated software for fault detection at solar thermal plants exists yet. Lacking an alternative, operators often use supervisory control and data acquisition (SCADA) applications. However, mainly targeted at industrial processes, they do not support analyzing the multidimensional sensor data and provide little functionality for exploring automatic fault detection results. While applications related to anomaly detection do exist for other domains, approaches so far either focus on different data like networks,[4–6] repeatable processes,[7–9] or traffic,[10] do not support the detailed analysis of multivariate dependencies,[11–13] or don't allow to access contextual information about the plant operation.[14–16] The only exception is MTV,[17] which was developed in parallel to this work, focusing on multivariate time series. However, it does not allow for analyzing correlations between faults.

Hence, this work studies the topic of visual fault detection for solar-thermal plants. Our main contributions are the following:

- We present a requirement analysis for visual fault detection at solar-thermal systems based on inquiries with domain experts, describing data and related tasks.
- Based on the requirements, We introduce the visual framework called SunScreen. The application allows to analyze detected anomalies, access manually annotated information, and explore the sensor data to support the fault detection process.
- We evaluate SunScreen using a usability test, compare our results to MTV and discuss why the current design of SunScreen does not yet meet all identified requirements.

XXXX-XXX © 2023 IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

FIGURE 1. Picture of a solar-thermal plant located in Graz, Austria, providing heat for the local district heating network. Copyright: SOLID Solar Energy Systems GmbH.

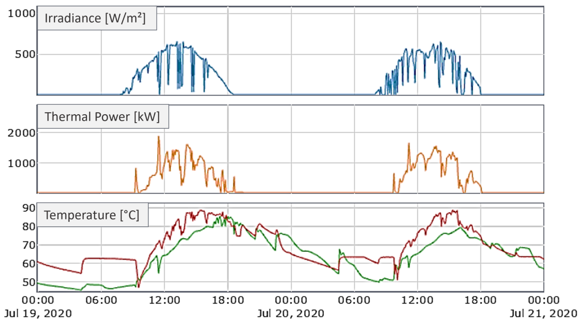


FIGURE 2. Example measurement data displaying irradiation, thermal yield, and inlet and outlet temperatures of collectors. Note how solar-thermal yield is correlated to irradiation but cannot be interpreted without considering time-dependencies due to heat up and correlations with other measurements.

DOMAIN

Solar-thermal plants are used for many applications, for example, providing heat for district heating networks, domestic hot water, industrial processes, desalination, or powering chiller for cooling. Plants are often closely designed to fit the customer's needs, leading to unique system designs. Especially large-scale plants are usually thoroughly monitored to ensure proper operation and react to faults quickly.

The challenge with solar-thermal data is its complex nature: The heat generation primarily depends on solar irradiation, but the efficiency of the plant is also highly influenced by consumer demand and the temperatures within different parts of the system. As such, measurements often follow daily radiation patterns but can also be interrupted by changes in irradiation (e.g., clouds) or changes in consumer demand. Additional time dependencies arise through the fluid's low inertia and heat accumulating in storage tanks and collectors. Hence, the data is multidimensional (e.g., irradiation and temperatures influencing thermal power), time-dependent (e.g., daily radiation patterns; heat accumulating in storage tanks and collectors; the slow traversing speed of the fluid in pipes), and non-linear (e.g., pumps suddenly starting).

REQUIREMENT ANALYSIS

The requirement analysis was done working closely together with experts from SOLID Solar Energy Systems GmbH (SOLID). As designers and operators of solar-thermal plants, they monitor many large-scale plants worldwide. They kindly agreed to provide measurement data and participate in multiple feedback sessions over one year. Additional domain knowledge was gathered by the first author, who worked at the company part-time for four years and participated in one monitoring session.

Data Abstraction

The data sets used by the monitoring personnel can be categorized into three groups based on their source (ref Figure 3):

The basis for analyzing solar-thermal plants is the *sensor* data. Based on the data sets provided, around 100 sensors are used to keep track of the most important measurements at each plant. Stored as timestamp-value pairs, the measurement data (S2) allows insights into the plant's various processes at a current time. In addition, semantic information (S1), like the sensor name and the unit of measurement, is also needed to interpret the data. Especially the position of the sensor (S3) and the derived hierarchy and plant structure are essential to understanding the flow of the fluids. Thus, displaying the plant structure is central in most SCADA systems and other industry-related visualization approaches. In addition, there are also logical correlations (S4) to keep in mind, which are based on physical relationships or system-control set-points. For example, the flow temperatures might be controlled based on the current demand, forming a logical correlation between them. Unfortunately, system-control information is often hidden in technical description sheets and is rarely available.

Second, the monitoring personnel needs contextual information about known *events* influencing the sys-

		Dataset	Dataset Type	Attributes
Sensor	S1	Details	table	sensor name, unit of the measurements
	S2	Measurements	time-series	timestamp, measurement value
	S3	Position	network	positions of sensor (topological, or structure of plant)
	S4	Correlations	network	links between correlated sensors
Event	E1	Details	table	type, status, severity, title, description of the event
	E2	Intervals	table	start, end of event
	E3	Affected sensors	list	list of sensors that are affected by event
	E4	Referenced sensors	list	list of sensors useful for understanding event
	E5	Relations	network	links between events, type of relation
Anomaly Algorithm	A1	Details	table	ID, class of algorithm, description
	A2	Intervals	table	start, end
	A3	Target	list	sensor(s) targeted by the algorithm
	A4	Input Features	network	sensor(s) used as input, importance of sensor
	A5	Probability	table	probability
	A6	Predictions	time-series	timestamp, predicted sensor values

FIGURE 3. Data sets needed for fault detection at solar-thermal plants, categorized into three groups. The terminology of Munzner[18] was used to determine the dataset type.

				Sensors				Events					Anomaly Algorithms					
				S1	S2	S3	S4	E1	E2	E3	E4	E5	A1	A2	A3	A4	A5	A6
Tasks		Questions:		Details	Measurements	Position	Correlations	Details	Intervals	Affected Sensors	Referenced Sensors	Relations	Details	Intervals	Target sensor	Input-Features	Probability	Predictions
T1	Overview	Q1	What is the layout of the system?	▨	▨	■	▨									▨		
		Q2	What is the overall status of the system?	▨	▨	▨		■	■	▨				■	▨	▨	▨	
		Q3	What events happened recently?					■	■	▨								
		Q4	Which sensors indicate anomalies? Are there any patterns?	▨	▨	▨		▨	▨	▨				■	■	▨	▨	
T2	Evaluate Anomalies	Q5	Which sensors are correlated to the anomaly?	▨		▨	▨							▨	■	■		
		Q6	What are the measurment values of affected sensors during the anomaly?	▨	■	▨	▨							■	▨	▨		
		Q7	How does the measurements differ from typical behaviour?	▨	■	▨	▨							■	▨	▨		■
		Q8	Is this a fault?	▨	■	▨	▨	▨	▨					▨	▨	▨	■	▨
		Q9	Do some anomalies correspond to the same fault?	▨	▨								▨	■	■	■		
		Q10	Which algorithms detected the anomalies?										■	▨	▨	▨		
		Q11	Is the anomaly detection algorithm working properly in gerneral?	▨	▨	▨	▨		▨	▨	▨		■	■	■	▨	▨	▨
T3	Fault Detection	Q12	Are there any abnormal patterns in the sensor data?	▨	■	▨	▨											
		Q13	Are there any events that indicate follow-up events?					■	■	▨	▨							
T4	Fault Diagnosis	Q14	Which sensors are affected by the fault?	▨	▨	▨	▨	▨		▨				▨	▨	▨		
		Q15	When did the event occur? Is it still active?		■			▨	▨					■				▨
		Q16	How did the fault evolve?		■	▨	▨	▨	▨	▨	▨	▨		■	▨	▨		▨
		Q17	What was the root-cause of the fault?	▨	■	▨	▨	▨	▨	▨	▨	▨		▨	▨	▨		▨
		Q18	Did the fault occur on sensor, component or system control level?	▨	▨	▨	▨	▨						▨	▨	▨		▨
		Q19	Is the fault related to other events?	▨	▨	▨	▨	▨	■	▨	▨	▨		■	▨	▨		
		Q20	Has this happend before? Is this a latent or reocurring fault?	▨	▨			■	▨	■		▨	▨		▨	▨		▨
		Q21	What is the severity of the fault?		▨	▨	▨	▨						▨	▨	▨		
T5	Annotate Events	Q22	How to document the event?	■	▨	▨	▨	▨					▨	■	■	▨		

FIGURE 4. Table showing high-level tasks and domain user questions in relation to the needed monitoring data identified in the data abstraction. Dark-colored boxes mark data essential to solving a question, while optional supporting data is marked in lighter colors. The information to draw this plot was gathered during discussions and sessions with the domain user and through feedback to rapid prototypes. However, distinctions between "essential" and "helpful" are somewhat vague and highly influenced by the author's opinion.

tem. For example, services conducted at a plant may explain abnormal sensor measurements, and known faults might help diagnose new anomalies. The data typically comprises a name and description of the event (E1) and a start and end time (E2). It also makes sense to reference which sensors are directly affected by the event (E3) and which are helpful for understanding the event (E4). In addition, events might also have relations to other events (E5). For example, some faults may cause other faults. These relations are needed to understand the evolution of a fault.

Finally, the monitoring personnel might also rely on *anomaly* detection algorithms to benefit from automation. The detected anomalies share similar characteristics to the event data, namely, semantic information, including a description of what the anomaly is about (A1) and a start and end time (A2). In addition, anomaly algorithms are often targeted at a specific sensor (A3) whose values are monitored and use multiple sensors as input (A4). Depending on the type of fault detection algorithm, the probability of the anomaly (A5) and predicted values for the target sensor (A6) might also be available. The main challenge with anomalies is the potentially high number of false alarms due to the complex nature of the plant behavior and the large number of sensors that could be tracked.[2,3]

Domain Tasks

We identified the following high-level tasks (T1-T5) during our discussions with the domain users. The summary can be seen in Figure 4, showing the identified tasks, related questions, and the data used to answer them.

Initially, the monitoring personnel needs an overview of the plant (T1) as a starting point for further investigation. The overview is required on two fronts: First, the user needs to understand the plant structure (Q1). This gives them enough context to correctly interpret the sensor data, anomalies, and events. On the other hand, the monitoring personnel wants to get a quick overview of the system status (Q2-4). By checking recent events and detected anomalies, the user might identify patterns or trends that need to be investigated first.

If any anomalies are found during T1, they need to be investigated (T2). In the end, the monitoring personnel needs to judge if the detected anomaly is indeed a fault or if it is just a false alarm due to rare operating conditions or inaccurate algorithms (Q8-Q11). Solving this task includes exploring correlated sensors and related events and checking the sensor data for further clues (Q5-Q7).

As automatic algorithms might not identify all faults, monitoring personnel must manually check the system for additional unidentified faults (T3). Here, monitoring personnel might look for any patterns in the sensor data that indicate abnormal system behavior (Q12). In addition, they might also look at known events, for example, if a fault is expected to have caused a secondary failure (Q13).

When a fault has been identified during T2 or T3, additional information about its cause and consequences must be derived as part of fault diagnosis (T4). Thus, the task is to identify the affected sensors and their location, root cause, evolution, and fault severity (Q14-Q21). Information is gathered from multiple angles of perspectives to describe the fault more clearly and highlight relations to other events.

Finally, information about events must be annotated (T5) to benefit from the derived knowledge (Q22). This annotation allows sharing the information with fellow analysts and system managers. In addition, the information about known events might be helpful in future monitoring sessions to explain anomalies, diagnose similar faults, or find root causes.

Design Goals

Additionally, the following requirements were identified for the visual application:

- *G1 Efficient workflow* —The focus is on efficiently directing the user to potential faults at the expense of undirected exploration. Otherwise, detecting and diagnosing faults will take too long, and monitoring will no longer be economically feasible.
- *G2 High coverage of faults* —The number of undetected faults must be as small as possible, and all critical faults must be detectable.
- *G3 Exportable results* —It must be possible to import and export events, for example, in an external database.
- *G4 Scalability* —The application should work for any arbitrary solar-thermal plant, independent of its size and structure. Having multiple hundreds of sensors per system should be supported.

RELATED WORK

Solar-Thermal domain

To the author's knowledge, there is no visual application dedicated to fault detection for solar-thermal plants. The only exception is *SunReports*,[16] which provides a basic line chart of sensor values and benchmarks for different time intervals. However, they do not use automated fault detection, and their product does not scale to hundreds of sensors (goal G4).

Missing other options, monitoring personnel typically rely on system control and data acquisition (SCADA) software. These applications allow them to check sensor values and control the system remotely. They usually feature a topological view of the sensor positions, their current measurements, and line charts of selected sensor measurements. Some software even provides custom alarms and annotations as well. However, being targeted at industrial processes generally, they lack the support for analyzing multivariate time-series data and integrating more complex anomaly detection algorithms.

Photovoltaic domain

While not much commercial software is available for the solar-thermal domain, many more fault-detection applications are available for photovoltaic (PV) solar systems. Automatic fault detection is typically done by checking for basic faults like missing data or comparing expected with measured power generation. The sensor data is shown via line charts, with colored backgrounds for periods where faults occurred, sometimes showing a summary description.

While most of these applications cover many identified tasks and data sets, they cannot be used for the solar-thermal domain. One reason is that the underlying data of solar-thermal applications is harder to interpret than PV. This complexity may be explained by the higher inertia of the solar-thermal fluid and the higher complexity of many factors (like irradiation, consumer demand, and storage capacity) mutually influencing the system's temperatures. Thus, many more sensors must be shown simultaneously to understand the data, and the sensors' position seems more relevant, too. In addition, the automatic algorithms for the solar-thermal domain are much more prone to false alarms.[2,3]

Design Studies

Applications to support fault detection have also been studied in various other domains. An overview is shown in Figure 5.

Huovinnen and Hietanen[7] focus on data from a pulp dry facility. They assume that most of their data corresponds to "normal" operating points and try to find these modes using clustering. Outliers that do not fit into these clusters are then regarded as anomalies. However, this approach is likely to fail in the case of solar-thermal data. In contrast to industrial processes, where measurements are typically stable and repeatable, the measurements of solar-thermal plants vary considerably due to changes in irradiation, consumer demand, and internal temperatures. As a result, the same operating conditions rarely occur twice, increasing the difficulty of distinguishing faults from normal operating conditions. A similar approach is also used by TripMiner by Riverio et al.,[10] which focuses on road traffic data to detect accidents or near-accidents using the same clustering approach.

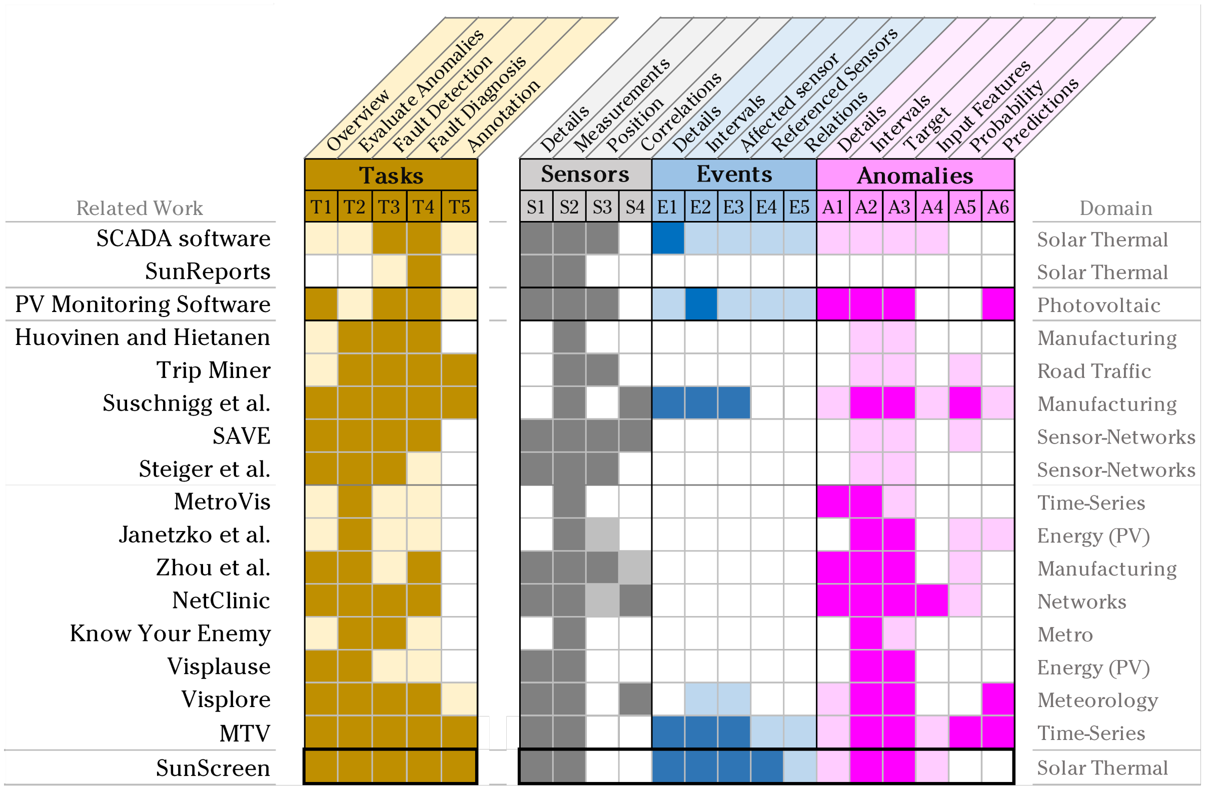
Another example is Suschnigg et al.,[8] which also focuses on manufacturing. Using different anomaly detection algorithms, they support users in detecting, analyzing, and annotating abnormal process cycles in manufacturing. While they cover almost all identified tasks and datasets, their approach requires repeatable processes. Similar to TripMiner and Huovinnen et al., this requirement is not met in the case of solar-thermal heat generation.

SAVE from Shi et al.[4] focuses on sensor networks and supports detecting routing problems between sensors. To do so, they visualize the routing paths, the correlations between sensor measurements and status, and the temporal evolution of routing paths and measurements. The user can then check the graphs for patterns and correlations manually. One drawback is that anomaly detection is not automated, so users must explore the data themselves. Thus, this might conflict with goal G1 Efficient Workflow.

Another example is Steiger et al.,[5] which tries to detect anomalies of power consumption in an energy grid. They use clustering to group similar daily patterns of each sensor. To explore the results, their visual framework shows the clusters in a scatter plot, positioning similar patterns next to each other. One limitation is that they require uni-variate data and that sensors measure the same quantity only.

MetroVis from Eichmann et al.[11] supports data scientists in evaluating their fault detection algorithms. The results are displayed using line charts, and some functionality is provided to compare different algorithm results. However, MetroVis only supports showing one sensor at a time, which is insufficient to understand the complex behavior of solar-thermal systems or diagnose anomalies.

Janetzko et al.[12] focus on anomaly detection for power consumption data. A tree map is used to display the hierarchy of the sensors. The sensor data is displayed in each panel using spiral graphs, calendar views, or line charts, while anomalies are displayed using color encoding. However, the tree-map layout does not scale to the hundreds of sensors used at solar-thermal systems violating goal G4 Scalability.



Related Work	Tasks					Sensors				Events					Anomalies						Domain
	Overview	Evaluate Anomalies	Fault Detection	Fault Diagnosis	Annotation	Details	Measurements	Position	Correlations	Details	Intervals	Affected sensor	Referenced Sensors	Relations	Details	Intervals	Target	Input Features	Probability	Predictions	
	T1	T2	T3	T4	T5	S1	S2	S3	S4	E1	E2	E3	E4	E5	A1	A2	A3	A4	A5	A6	
SCADA software	▨	▨	■	■	▨	■	■	■		■	▨	▨	▨	▨	▨	▨	▨	▨			Solar Thermal
SunReports			▨	■		■	■														Solar Thermal
PV Monitoring Software	■	▨	■	■	▨	■	■	■		▨	■	▨	▨	▨	■	■	■			■	Photovoltaic
Huovinen and Hietanen	▨	■	■	■			■									▨	▨				Manufacturing
Trip Miner	▨	■	■	■	■		■	■								▨	▨		▨		Road Traffic
Suschnigg et al.	■	■	■	■	■		■		■	■	■	■			▨	■	■	▨	■	▨	Manufacturing
SAVE	■	■	■	■		■	■	■	■							▨	▨		▨		Sensor-Networks
Steiger et al.	■	■	■	▨		■	■	■								▨	▨				Sensor-Networks
MetroVis	▨	■	▨	▨			■								■	■	▨				Time-Series
Janetzko et al.	▨	■	▨	▨			■	▨								■	■		▨	▨	Energy (PV)
Zhou et al.	■	■	▨	■		■	■	■	▨						■	■	■		▨		Manufacturing
NetClinic	■	■	■	■		■	■	▨	■						■	■	■	■	▨		Networks
Know Your Enemy	▨	■	■	▨			■									■	▨				Metro
Visplause	■	■	▨	▨		■	■									■	■				Energy (PV)
Visplore	■	■	■	■	▨	■	■		■		▨	▨			▨	■	■			■	Meteorology
MTV	■	■	■	■	■	■	■			■	■	■	▨	▨	▨	■	■	▨	■	■	Time-Series
SunScreen	■	■	■	■	■	■	■			■	■	■	■	▨	▨	■	■	▨			Solar Thermal

FIGURE 5. Table showing which tasks and data sets are supported by related work. The frameworks support tasks and data shown in darker colors, while lighter colors indicate that they are only partially supported. If the task or data is not supported, the cell is colored white.

Zhou et al.[9] focus on monitoring a large manufacturing facility. A rule-based method assigns a criticality index (i.e., anomaly score) to each part of the system for its current-, short-term-, and long-term behavior. The data shows the system's layout, with components colored based on their anomaly score. The user thus gets an intuitive overview (T1) while anomalies can be evaluated (T2) and further diagnosed (T3) in subsequent views. The authors chose a color encoding to visualize the temperature variations compared to its reference for each zone. However, this requires that constant reference temperatures are available. Unfortunately, such an encoding is impossible for solar thermal data, as temperatures fluctuate depending on radiation and demand. In addition, details of only one zone can be shown at a time, while analysis of solar thermal applications requires information on multiple parts of the system simultaneously.

Another inspiring example is NetClinic from Lui et al.[6] They aim to support fault diagnosis for computer networks, using the results from an external diagnosis tool. Conveniently, the tool also provides suggestions for potential culprits (T4). The data is displayed using a network graph, showing machines and their components in circular hierarchical nodes and their correlations as links between nodes. Another view contains the list of anomalies, highlighting the path from the culprit to the affected machine on click. Unfortunately, the visual framework cannot explore the temporal evolution of the data. While this is not critical to diagnose faults in computer networks (e.g., wrong router/firewall configurations), solar thermal data cannot be understood without a temporal context. In addition, no comparable diagnosis algorithm is available for solar thermal applications yet.

The primary goal of KnowYourEnemy from Gschwandtner et al.[13] is to check the data for quality issues. Users can either rely on automatic checks (T2) or investigate the data themselves (T3). Results are displayed using a heat map, which can be configured to show combinations of measurements, anomaly scores, and time in different granularities. However, the heat map can only show a maximum of three sensors at a time, while many more correlations might be needed to understand specific behaviors of solar-thermal systems.

Visplause[14] also focuses on data quality assessment. An overview of anomalies versus time is provided using a heat map. Each lane can be extended to show sensors and algorithm types while time is shown on the x-axis. The size of the cells depicts the number of anomalies found for each period, and the color depicts the type of anomaly algorithm that found the anomaly. Selective sensor measurement values are displayed redundantly using a table and a line chart. This visual framework is considered very similar to SunScreen. However, tasks like annotating faults (T5) or manual fault detection (T3) are not directly supported. As a result, no contextual information can be accessed during diagnosis, and exploring results seems to focus on one sensor at a time only.

Some of these issues were resolved by the more advanced Visplore,[15] developed by the same authors. For example, checking the details of multi-variate data is possible by displaying multiple sensor measurements in subplots next to each other. Recently, a labeling functionality was added to the software as well. In addition, many other functionalities like correlation analysis and different data representations further allow exploring the data to a greater extent. However, the extensive functionality conflicts with goal *G1 Efficient workflow*, as users might get distracted or overwhelmed by the sheer number of potential choices for displaying the data. In addition, the labeling does not allow access to detailed information about the events (E1) yet.

In summary, the existing visual frameworks above do not fulfill all the requirements for fault detection at solar-thermal plants. While some applications focus on data with different properties, especially annotation T5, and showing event data is rarely supported (see Figure 5).

The exception is MTV,[17] developed by Liu et al. in parallel to this work. They focus on the detection and annotation of anomalies in multivariate time series. The identified tasks, goals, and the used datasets are very similar to the ones defined in the Requirement Analysis section. Their tool allows the creation of "pipelines" running anomaly detection algorithms. An overview page displays summarized results for each pipeline, while a detailed page shows the measurement data and anomalies using line charts, circular graphs, and highlighting. A special emphasis is put on labeling and collaborative annotation of the anomalies. The authors report great feedback from users from the aerospace and energy domain. The only limitations reported by the authors are a lack of support for a large number of measurements and no support for comparing adjacent anomaly events. However, both these functionalities are important for successfully monitoring solar-thermal plants (Q4-7, Q19).

SunScreen

Hence, we present the visual framework SunScreen, which was designed explicitly for visual fault detection at solar thermal plants. It was developed using rapid prototyping based on the results from the requirement analysis.

Implementation Details

SunScreen is implemented in vue.js, primarily relying on the libraries D3.js and Plotly.js. SOLID provides data, including almost all the data sets identified in the data abstraction. Hence, the tool can access events manually inserted by the monitoring personnel (E1-E4) and the sensor data (S1-S3) from about 100 sensors per plant. However, relations between events (E5) and sensor correlations (S4) were unavailable. Finally, an anomaly database provides results from automatic fault detection algorithms (A1-A5). More specifically, the following algorithm types are used:

- *Missing data* —Checks if sensor data is missing or contains NAN-values (i.e., Not A Number).
- *Constant data* —Checks if sensor values stay constant for too long.
- *Collector stagnation* —Checks whether the collector temperature exceeds 130℃.
- *SunSearcher* — Early version of a machine learning algorithm called Fault-Detective.[3] It automatically identifies correlated sensor measurements using recursive feature selection and then models the relations using random forest regression. Multiple algorithms are used to track all sensor measurements. An anomaly is raised if the difference between predicted and measured values is too high.

Design

A screenshot of SunScreen can be seen in Figure 6. Instead of describing each part of the framework with words, we present the design of SunScreen using a Data-Comic in Figure 7. This idea is inspired by Bach et al.,[19] and we hope to thereby reduce the amount of switching between text and figure. In addition, the Data Comic in Figure 8 shows how SunScreen can detect and diagnose faults.

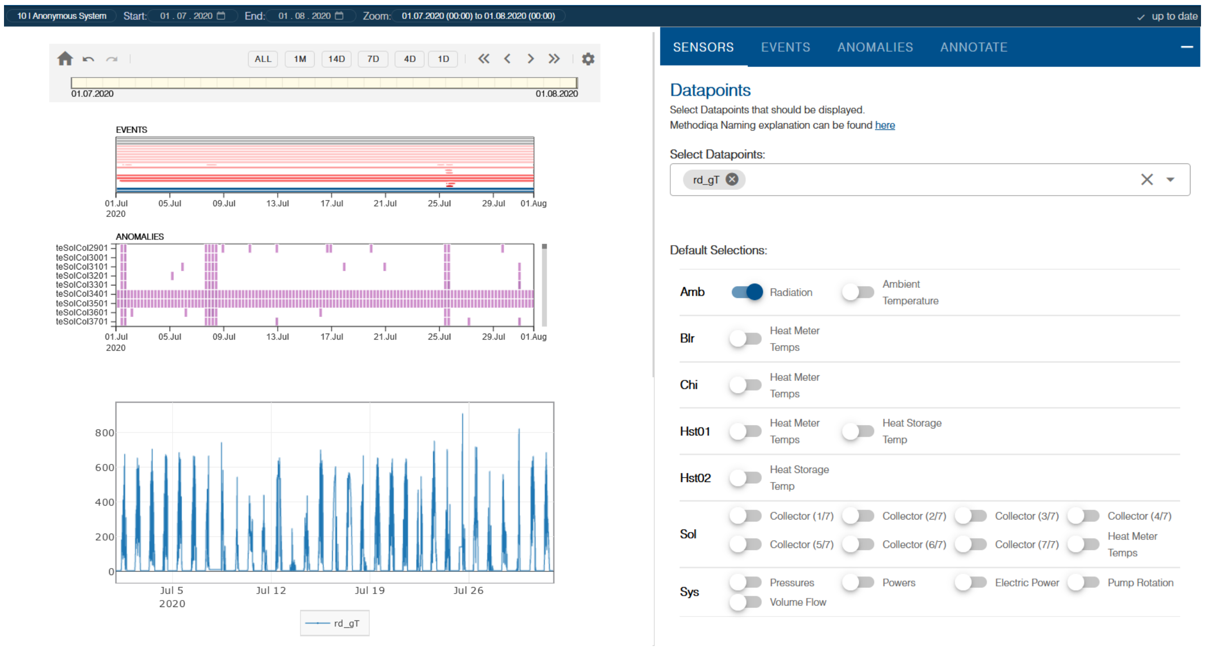


FIGURE 6. Screenshot of SunScreen. A description of each part can be seen in Figure 7.

Design Decisions

SunScreen was developed using rapid prototyping, and many feedback sessions with domain users were carried out to improve its design. This subsection discusses these results and explains why certain visual encodings were used for certain parts of SunScreen.

Why focus on time? The central part of SunScreen shows the data (events, anomalies, and sensor values) versus time. However, we also experimented with other options in the early stages of the prototype. For example, using the position of the sensors (displayed as a topological view), the relations between sensors (displayed as graphs), or the sensors and their anomaly score (displayed as a list) as the main view. Nevertheless, the domain users preferred the time-based approach, arguing that understanding the system's temporal behavior is the data's most critical aspect. For example, latent faults may only be active at certain times, and detecting and diagnosing faults without analyzing the evolution of measurement values is impossible.

Why not use the sensor positions? The sensor positions are vital to comprehending the system's behavior (see Data Abstraction S4). They play a critical role in interpreting the sensor data and have been identified as vital to gaining an overview of the system (T1). Nevertheless, the sensor positions are currently not used by SunScreen. The reason is that they do not change over time; thus, a simple printout or image of the system hydraulics is sufficient to support most of the identified tasks. Although this is only an interim solution, using the sensor positions was postponed, and more time was spent on features with higher priority instead.

Why show anomalies and events in different graphs? Although the data properties of anomalies and events are very similar, we show them in different subplots. The reason is that events are secured information from the domain user, while the detected anomalies must be verified first. The difference also shows in the relationship to sensors: Anomalies are tightly connected to sensors, as their measurement values are used for fault detection. In contrast, events might not be connected to sensors at all (e.g., in the case of a non-intrusive service), or affected sensors might not have been identified yet. Thus, the decision was to show events and anomalies in separate graphs.

Why use Gantts to display events? The event data is displayed using Gantt charts—encoding the event's start, end, type, and severity. Other early ideas used line charts or heat maps to show the number of events or severity versus time. However, these approaches were ruled out because the domain user needs information about each event instead of aggregated

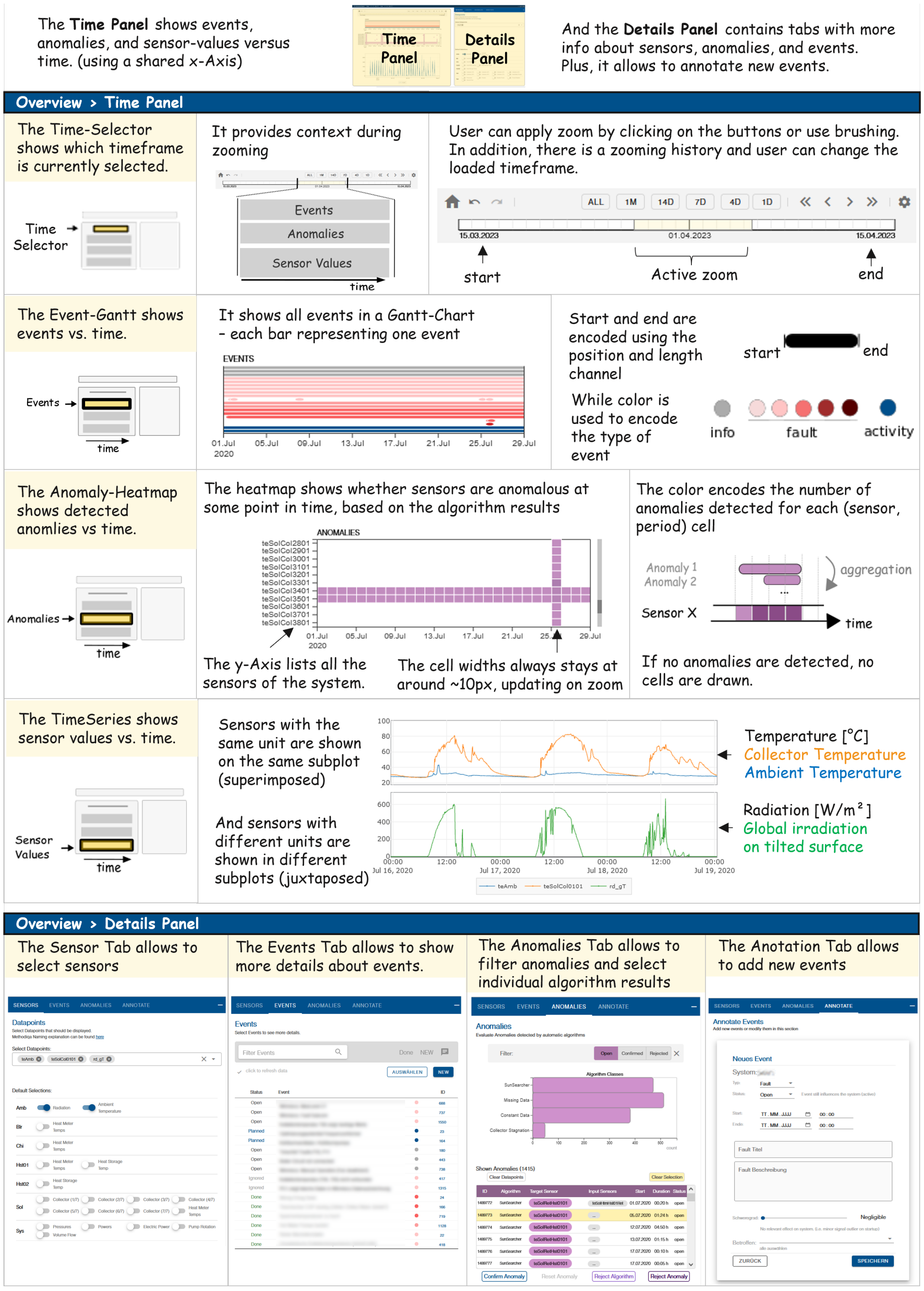
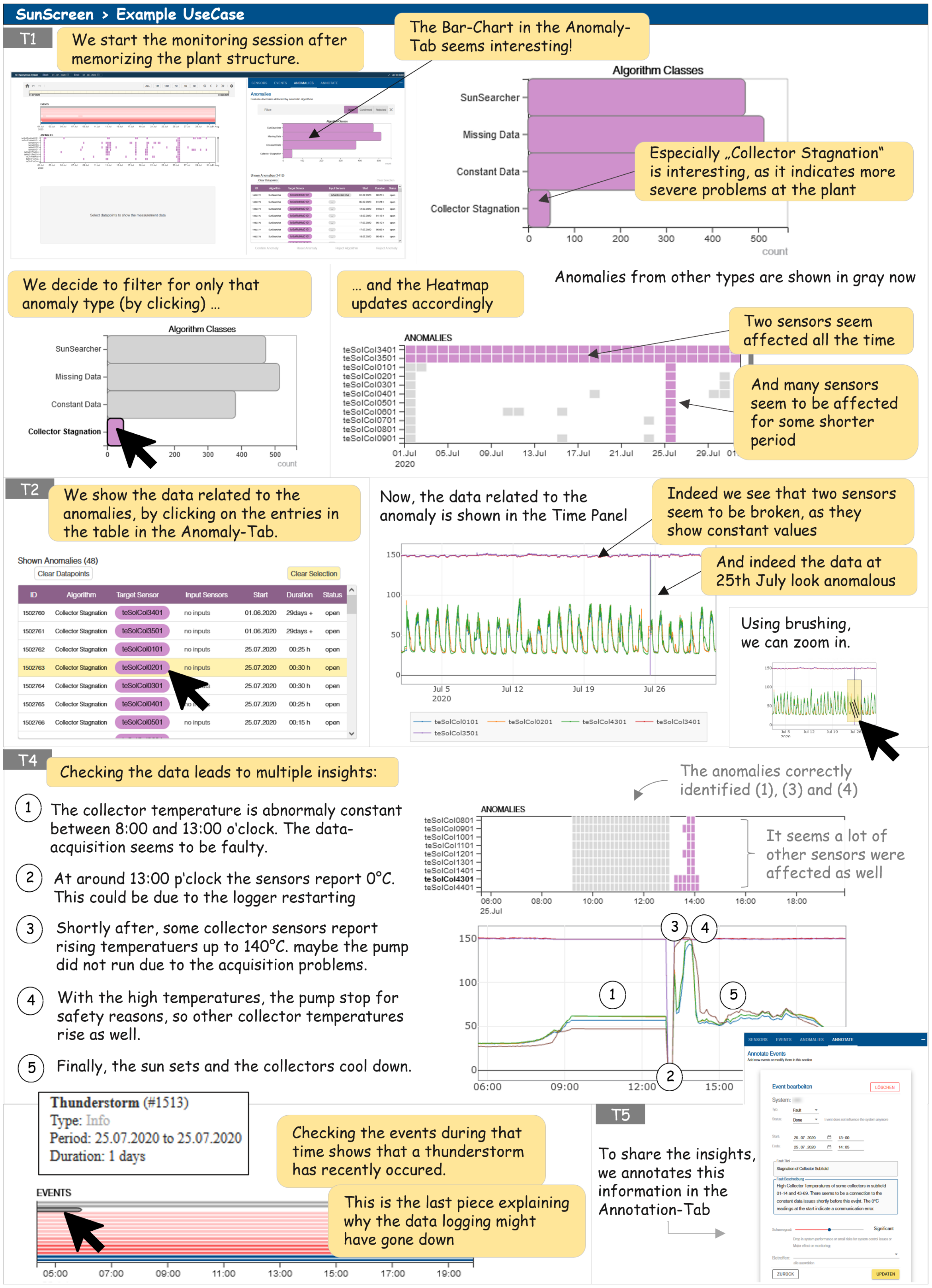

The Time Panel shows events, anomalies, and sensor-values versus time. (using a shared x-Axis)
Time Panel
Details Panel
And the Details Panel contains tabs with more info about sensors, anomalies, and events. Plus, it allows to annotate new events.
Overview > Time Panel
The Time-Selector shows which timeframe is currently selected.
Time Selector
It provides context during zooming
Events
Anomalies
Sensor Values
time
User can apply zoom by clicking on the buttons or use brushing. In addition, there is a zooming history and user can change the loaded timeframe.
ALL 1M 14D 7D 4D 1D
15.03.2023
01.04.2023
15.04.2023
start
Active zoom
end
The Event-Gantt shows events vs. time.
Events
time
It shows all events in a Gantt-Chart – each bar representing one event
EVENTS
01.Jul 2020
05.Jul
09.Jul
13.Jul
17.Jul
21.Jul
25.Jul
29.Jul
Start and end are encoded using the position and length channel
start
end
While color is used to encode the type of event
info
fault
activity
The Anomaly-Heatmap shows detected anomlies vs time.
Anomalies
time
The heatmap shows whether sensors are anomalous at some point in time, based on the algorithm results
ANOMALIES
teSolCol2801
teSolCol2901
teSolCol3001
teSolCol3101
teSolCol3201
teSolCol3301
teSolCol3401
teSolCol3501
teSolCol3601
teSolCol3701
teSolCol3801
01.Jul 2020
05.Jul
09.Jul
13.Jul
17.Jul
21.Jul
25.
29.Jul
The y-Axis lists all the sensors of the system.
The cell widths always stays at around ~10px, updating on zoom
The color encodes the number of anomalies detected for each (sensor, period) cell
Anomaly 1
Anomaly 2
aggregation
Sensor X
time
If no anomalies are detected, no cells are drawn.
The TimeSeries shows sensor values vs. time.
Sensor Values
time
Sensors with the same unit are shown on the same subplot (superimposed)
And sensors with different units are shown in different subplots (juxtaposed)
Temperature [°C]
Collector Temperature
Ambient Temperature
Radiation [W/m²]
Global irradiation on tilted surface
00:00 Jul 16, 2020
12:00
00:00 Jul 17, 2020
12:00
00:00 Jul 18, 2020
12:00
00:00 Jul 19, 2020
teAmb
teSolCol0101
rd_gT
Overview > Details Panel
The Sensor Tab allows to select sensors
The Events Tab allows to show more details about events.
The Anomalies Tab allows to filter anomalies and select individual algorithm results
The Anotation Tab allows to add new events
SENSORS
EVENTS
ANOMALIES
ANNOTATE
Datapoints
Select Datapoints
Default Selections
Events
Filter Events
AUSWÄHLEN
NEW
Anomalies
Filter:
Open
Confirmed
Rejected
Algorithm Classes
SunSearcher
Missing Data
Constant Data
Collector Stagnation
Shown Anomalies (1415)
Confirm Anomaly
Reset Anomaly
Reject Algorithm
Reject Anomaly
Annotate Events
Neues Event
Fault Titel
Fault Beschreibung
Negligible
ZURÜCK
SPEICHERN

FIGURE 7. Data Comic providing an overview of SunScreen


SunScreen > Example UseCase
T1
We start the monitoring session after memorizing the plant structure.
The Bar-Chart in the Anomaly-Tab seems interesting!
Algorithm Classes
SunSearcher
Missing Data
Constant Data
Collector Stagnation
count
Especially „Collector Stagnation" is interesting, as it indicates more severe problems at the plant
We decide to filter for only that anomaly type (by clicking) …
… and the Heatmap updates accordingly
Anomalies from other types are shown in gray now
ANOMALIES
Two sensors seem affected all the time
And many sensors seem to be affected for some shorter period
T2
We show the data related to the anomalies, by clicking on the entries in the table in the Anomaly-Tab.
Now, the data related to the anomaly is shown in the Time Panel
Indeed we see that two sensors seem to be broken, as they show constant values
And indeed the data at 25th July look anomalous
Using brushing, we can zoom in.
teSolCol0101
teSolCol0201
teSolCol4301
teSolCol3401
teSolCol3501
T4
Checking the data leads to multiple insights:
1 The collector temperature is abnormaly constant between 8:00 and 13:00 o'clock. The data-acquisition seems to be faulty.
2 At around 13:00 p'clock the sensors report 0°C. This could be due to the logger restarting
3 Shortly after, some collector sensors report rising temperatuers up to 140°C. maybe the pump did not run due to the acquisition problems.
4 With the high temperatures, the pump stop for safety reasons, so other collector temperatures rise as well.
5 Finally, the sun sets and the collectors cool down.
The anomalies correctly identified (1), (3) and (4)
It seems a lot of other sensors were affected as well
Thunderstorm (#1513)
Type: Info
Period: 25.07.2020 to 25.07.2020
Duration: 1 days
Checking the events during that time shows that a thunderstorm has recently occured.
EVENTS
This is the last piece explaining why the data logging might have gone down
T5
To share the insights, we annotates this information in the Annotation-Tab
SENSORS
EVENTS
ANOMALIES
ANNOTATE
LÖSCHEN
ZURÜCK
UPDATEN

Shown Anomalies (48)

Clear Datapoints | Clear Selection

ID	Algorithm	Target Sensor	Input Sensors	Start	Duration	Status
1502760	Collector Stagnation	teSolCol3401	no inputs	01.06.2020	29days +	open
1502761	Collector Stagnation	teSolCol3501	no inputs	01.06.2020	29days +	open
1502762	Collector Stagnation	teSolCol0101	no inputs	25.07.2020	00:25 h	open
1502763	Collector Stagnation	teSolCol0201	no inputs	25.07.2020	00:30 h	open
1502764	Collector Stagnation	teSolCol0301	[illegible]	25.07.2020	00:30 h	open
1502765	Collector Stagnation	teSolCol0401	[illegible]	25.07.2020	00:25 h	open
1502766	Collector Stagnation	teSolCol0501	no inputs	25.07.2020	00:15 h	open

Annotate Events
Add new events or modify them in this section

Event bearbeiten

System:
Typ: Fault
Status: Done — Event does not influence the system anymore
Start: 25.07.2020 13:00
Ende: 25.07.2020 14:05
Fault Titel: Stagnation of Collector Subfield
Fault Beschreibung: High Collector Temperatures of some collectors in subfield 01-14 and 43-69. There seems to be a connection to the constant data issues shortly before this event. The 0°C readings at the start indicate a communication error.
Schweregrad: Significant — Drop in system performance or small risks for system control issues or Major effect on monitoring;
Betroffen: alle auswählen

FIGURE 8. Data Comic showing a exemplary use of SunScreen

information. In contrast, the Gantt chart provides the user with exactly the information they need. Namely, the start (start of the Gantt), end (end of the Gantt), duration (length of the Gantt), type (hue of the Gantt), severity (saturation), and the title of the event (embedded as tool-tip). By positioning the Gantts on lanes based on the type and severity of the event, the visual clutter is minimized.

Why use a heat map to display anomalies? The Anomaly-Heat-map supports the user in identifying anomalous sensors and detecting correlations between anomalies. Especially the temporal evolution of the anomalies is essential to the user. Hence, the graph should encode the number of anomalies of each sensor versus time. Using heat maps to display this information is thus both practical and intuitive, as they encode time (x-axis), sensor names (y-axis), and the number of anomalies (color channel) intuitively. This choice is backed up by Visplause[14] and Visplore,[15] which successfully use a similar encoding and inspired our Anomaly-Heat-map. Alternatives like line charts and horizon graphs were ruled out because they put too much emphasis on the number of anomalies and introduce unnecessary clutter. At the same time, Gantt graphs require too much space.

Why mix juxtaposition and superposition in the line charts? In the Time Panel, measurement values of sensors with the same unit are shown in the same subplot, while sensors with different units are shown in separate subplots. This was done to limit visual clutter and better support comparisons. For example, a user might want to select all collector temperatures and check for outliers. Due to the high correlations, the measurements are typically very similar to each other. Detecting outliers is thus very easy if values are shown on the same plot. In contrast, the lines are hard to distinguish if too much different data is shown. Hence, the choice was made to separate sensor data with different units, as they often have different trends, and detailed comparisons are less likely.

Why Tabs for the Details Panel? The requirement analysis shows that not all parts of the data are needed at once. Instead, solving different tasks might require very different parts of the data. Showing all the data together would need a lot of space and overwhelm the user with information. Thus, tabs enable details on demand while hiding not-required information. This way, the user can act flexibly but still focus on the data they need.

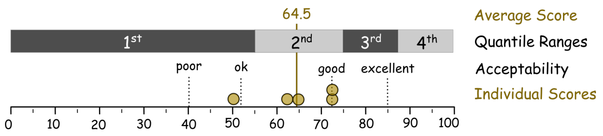


FIGURE 9. System Usability Scale results for SunScreen. The scores of each participant are displayed as yellow circles, while the average value is displayed as a yellow line. In addition, quantile ranges and acceptability for SUS tests based on empirical studies[21] are displayed to better interpret the results

Evaluation

Usability tests were conducted with five domain experts from SOLID. All participants (two researchers, two engineers, and one target user) are experienced in analyzing solar-thermal data, typically using SCADA systems.

After a short introduction to SunScreen (about 20 minutes long), participants were asked to perform some predefined tasks or explore SunScreen on their own, based on their personal preference. To gain more insights, the participants were told to describe their interactions and what they intended to do by speaking aloud. After working with SunScreen for one hour, they were asked to fill out an unmodified System-Usability-Scale (SUS) questionnaire.[20,21] In the end, the participants were asked to give overall feedback about SunScreen in a short semi-structured interview (about 5 minutes).

The results from the SUS questionnaires can be seen in Figure 9. Individual scores range between 50 and 72.5, and the average score is 62.5. Based on empirical studies done by Bangor et al.,[21] this indicates only moderate usability of SunScreen. Statements that received low scores were “I think I would need the support of a technical person to be able to use this system” and “I would imagine that most people would learn to use this system very quickly.” All other statements were rated as good or at least with average scores by all participants.

Even though the individual scores for each participant differed, the feedback from the interview was relatively homogeneous. Apart from some bugs and inconsistencies that the participants discovered, the following issues are the most relevant:

- *Missing knowledge / “overwhelming”* — Most users reported being overwhelmed with information at some point in their interaction with SunScreen: “Too much information at once . . . events, anomalies . . . one cannot solve this in one session". One domain user remarked that "the problem is not the tool, but that there are very few people that can interpret both the sensor data and the algorithm results." These types of anomaly detection algorithms are still new to most domain users who participated in the usability study. Hence, being tasked to understand the algorithm results and interpreting the system's behavior simultaneously was a difficult task (see, for example, Figure 11). It indicates that the tool should provide better guidance and overviews for the user and better ways to make the algorithms understandable. Another user mentioned that "one has to grow into it," indicating that more extensive training might also result in better usability of SunScreen.

› *"Too many anomalies"* — Another issue was the sheer number of anomalies found by the algorithms. Because they were not optimized, the algorithms were very sensitive, resulting in many wrongly identified anomalies (false positives). This was done deliberately to check if SunScreen is good enough to compensate for bad fault detection algorithms. However, users were frustrated when faced with thousands of anomalies and an extraordinary rate of incorrect suggestions (around 70%-90% false positives). They reported that "There are thousands of useless anomalies, and only maybe a hundred of them are correct. I don't have enough time to check them all" (see, for example, Figure 10). Hence, SunScreen cannot compensate for such a high false-positive rate of anomalies. Instead, the anomaly algorithms need better accuracy, or SunScreen must be improved to identify and filter out incorrect anomalies.

However, a user also highlighted the potential of SunScreen, arguing that it is nice to combine automatic fault detection results, annotated events, and sensor data in one tool. Mainly the direct annotation of events was regarded very positively.

CONCLUSION

SunScreen is the first visual framework for fault detection directly targeted at solar-thermal systems. It allows for analyzing the complex multi-variate system data, evaluating automatically detected anomalies, and viewing system events which give vital hints to interpret the data. By annotating discovered events, SunScreen allows saving the information for future sessions. The use case clearly shows evidence that SunScreen is usable. At the same time, the feedback from the participants emphasizes that integrating system events and supporting annotation is an integral first step to improving the quality of monitoring. However, the usability test shows that the tool does not yet fulfill the user's requirements.

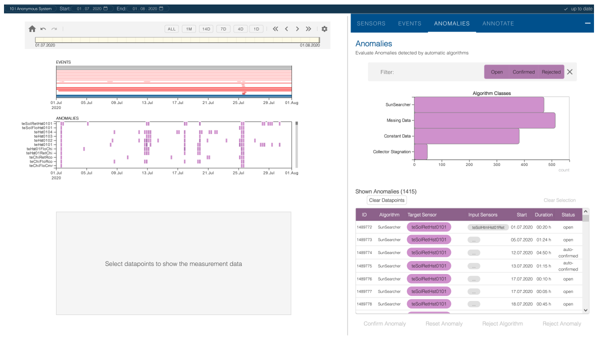


FIGURE 10. Starting the application, the users are confronted with a very large number of anomalies (in this case, 1415). Most of them are false positives.

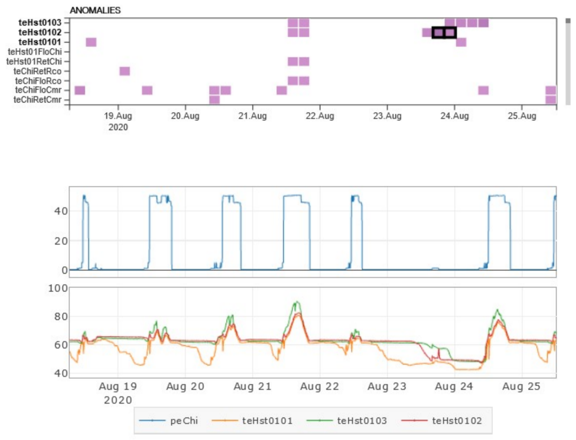


FIGURE 11. The graph shows measurement data related to the anomaly highlighted in the Anomaly heat map. Raised by a SunSearcher algorithm, the anomaly is likely caused by a rainy day resulting in unusually low-temperature readings (see lower plot) on the 24th of August. However, as this is typical behavior to the domain experts, it was not clear to them what the anomaly suggested.

Encountered Pitfalls

Reflecting on the evaluation results, we believe to have experienced the pitfall "PF-4: No Real Data Available" as described by Sedlmair et al.[22] The anomaly detection algorithms were introduced shortly before the development of SunScreen started. Thus, the algorithms lacked accuracy, and the domain users were not yet accustomed to interacting with these new anomalies and algorithms. As a result, both the requirement analysis and the evaluation of SunScreen are less accurate. First, observing the user carrying out the Evaluate Anomaly task (T2) during the requirement engineering was impossible. Second, feedback to SunScreen is mixed up with feedback to the anomaly detection algorithm SunSearcher.[3] In hindsight, a visual framework focusing on manual fault detection would have been easier to validate. Nevertheless, it was a strategic choice of the stakeholders to opt for automatic fault detection to benefit from the automation.

Design Goals

As a result of this pitfall, the design goals are only partially fulfilled. *G1 Efficient workflow* is inhibited by the number of anomalies detected. For the same reason, *G2 High coverage of faults* is also not met, as it is impossible to evaluate all identified anomalies. On the other hand, we regard the design goals *G3 Exportable results* and *G4 Scalability* as success. The annotation received positive feedback, and no concerns were reported about using the tool for different, more complex solar-thermal systems.

Comparison to MTV

Developed in parallel, there are some similarities between SunScreen and MTV.[17] The goals, identified tasks, and available datasets are practically the same. Both frameworks use regression-based machine-learning algorithms to detect anomalies, let users analyze the results, and put particular emphasis on annotating the results. The main view of both tools displays the measurements using line charts, and both tools use the outer right part of the screen for annotation and accessing more details.

Differences to SunScreen are especially the functionality for adding new pipelines (i.e., algorithms), displaying algorithm predictions and residuals, searching for similar measurements, and overall better usability. The former two functionalities significantly improve the explainability of the algorithm results. This indicates that utilizing this data in SunScreen might have helped users interact with the anomalies. In contrast, SunScreen emphasizes the time-dependency between events and anomalies and shows different algorithm results alongside each other. While, on the one hand, this is required for successful Fault-Diagnosis at solar-thermal systems (T4, Q19), it also confronts the user with a multitude of anomaly results that might overwhelm the user. Nevertheless, we believe that most of the differences in usability can be explained by the different accuracy of the anomaly detection algorithms used.

The similarities between the two frameworks also indicate that the design choices of SunScreen might be generalizable to multivariate time series in general, especially in the case of highly correlated measurements and in case temporal relations between events and anomalies are essential.

Future work

After the usability test, the choice was made to discard the anomaly-related views. Without the many anomalies, users were less overwhelmed, and SOLID adopted the tool for monitoring purposes. Future work focuses on continuously improving the visual framework and reintroducing more easy-to-understand and accurate anomaly algorithms. Other future work might contain:

- Adding more support for design goal *G1 Efficient workflow* by introducing new visual interfaces for getting an overview of anomalies, events, and sensor data.
- Showing anomaly algorithm predictions and results to improve the explainability of the anomaly algorithms and aid the user in decision-making.
- Reworking the annotation mechanism, using comments (E1) and relations between events (E5) similar to issue tracking systems and revising the Gantt-Chart to show the relations.
- Incorporating the sensor position and the structure of the plant (S4) might further improve the manual fault detection and the overall usability of SunScreen.

REFERENCES

[1] W. Weiss, and M. Spörk-Dür, "Solar Heat Worldwide," AEE Intec, Gleisdorf, Austria, IEA Solar Heating and Cooling Programme, 2022.

[2] C. Schmelzer, M. Georgii, C. Sauer, J. Orozaliev, and K. Vajen, "Fault detection for solar thermal systems: evaluation and improvement of existing algorithms," in *Proc. EuroSun2022*, 2022.

[3] L. Feierl, V. Unterberger, C. Rossi, B. Gerardts, and M. Gaetani, "Fault detective: Automatic fault-detection for solar thermal systems based on artificial intelligence," Solar Energy Advances, Volume 3, 2023, DOI: https://doi.org/10.1016/j.seja.2023.100033.

[4] L. Shi, Q. Liao, Y. He, R. Li, A. Striegel, and Z. Su, "SAVE: Sensor anomaly visualization engine," IEEE Conference on Visual Analytics Science and Technology (VAST), pp. 201-210, 2011.

[5] M. Steiger, J. Bernhard, S. Mittelstädt, H. Lücke-Tieke, D. Keim, T. May, and J. Kohlhammer, "Visual Analysis of Time-Series Similarities for Anomaly Detection in Sensor Networks," Computer Graphics Forum, vol. 33, pp. 401-410, 2014.

[6] T. Liu, B. Lee, S. Kandula, and R. Mahajan, "NetClinic: Interactive Visualization to Enhance Automated Fault Diagnosis in Enterprise Networks," IEEE Symposium on Visual Analytics Science and Technology, pp. 131-138, 2010.

[7] M. Huovinnen and V. Hietanen, "Monitoring and diagnosis of large scale industrial systems," in *Proc. IFAC*, vol. 39, no. 13, pp.831-836, 2006.

[8] J. Suschnigg, B. Mutlu, G. Koutroulis, V. Sabol, S. Thalmann, and T. Schreck, "Visual Exploration of Anomalies in Cyclic Time Series Data with Matrix and Glyph Representations," Big Data Research, vol.26, 16 August 2021, DOI: https://doi.org/10.1016/j.bdr.2021.100251.

[9] F. Zhou, X. Lin, X. Luo, Y. Zhao, Y. Chen, N. Chen, and W. Gui, "Visually enhanced situation awareness for complex manufacturing facility monitoring in smart factories," Journal of Visual Languages & Computing, vol. 44, pp. 58-69, February 2018.

[10] M. Riverio, M. Lebram, and M. Elmer, "Anomaly Detection for Road Traffic: A Visual Analytics Framework," IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 8, pp. 2260-2270, August 2017.

[11] P. Eichmann, F. Solleza, N. Tatbul, and S. Zdonik, "Visual Exploration of Time Series Anomalies with Metro-Viz," Proceedings of the 2019 International Conference on Management of Data, pp. 1901-1904, 25 June 2019.

[12] H. Janetzko, F. Stoffel, S. Mittelstädt, and D. A. Keim, "Anomaly detection for visual analytics of power consumption data," Computers & Graphics, vol. 38, pp. 27-37, 2014.

[13] T. Gschwandtner and O. Erhart, "Know Your Enemy: Identifying Quality Problems of Time Series Data," IEEE Pacific Visualization Symposium (PacificVis), pp. 205-214, 2018.

[14] C. Arbesser, F. Spechtenhauser, T. Muhlbacher, and H. Piringer, "Visplause: Visual Data Quality Assessment of Many Time Series Using Plausibility Checks," IEEE Transactions on Visualization and Computer Graphics, no. Vol.23, pp. 641-650, January 2017.

[15] M. Vuckovic and J. Schmidt, "Visual Analytics Approach to Comprehensive Meteorological Time-Series Analysis," Data, vol. 5, no. 4, 2020.

[16] SunReports, "SunReports," SunReports, 2011. [Online]. Available: http://www.sunreports.com/. [Accessed 13 November 2022].

[17] D. Liu, S. Alnegheimish, A. Zytek, and K. Veermachaneni, "MTV: Visual Analytics for Detecting, Investigating, and Annotating Anomalies in Multivariate Time Series," Proceedings of the ACM on Human-Computer Interaction, vol. 6, no. 103, pp. 1-30, April 2022, DOI: https://doi.org/10.1145/3512950

[18] T. Munzner, "Visualization Analysis & Design", Boca Raton, London, New York, CRC Press, 2014.

[19] B. Bach, N. H. Riche, S. Carpendale, and H. Pfister, "The Emerging Genre of Data Comics," IEEE Computer Graphics and Applications, pp. 6-13, May/June 2017.

[20] J. Brooke, "SUS: A 'Quick and Dirty' Usability Scale," in Usability Evaluation In Industry, London, Taylor & Francis Ltd, 1996, pp. 189-195.

[21] A. Bangor, P. T. Kortum, and J. T. Miller, "An Empirical Evaluation of the System Usability Scale," International Journal of Human–Computer Interaction, pp. 574-594, 30 July 2008.

[22] M. Sedlmair, M. Meyer, and T. Munzner, "Design Study Methodology: Reflections from the Trenches and the Stacks," IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 12, pp. 2431-2440, December 2012.

Lukas Feierl is a data scientist working at SOLID Solar Energy Systems GmbH, Graz, Austria. His current research interest is the monitoring of solar-thermal systems. He received his BSc in Physics at the University of Vienna and is currently studying Computation Science focusing on visual data analytics. Contact him at l.feierl@solid.at.

Torsten Möller is the head of the Visualization and Data Analysis research group at the Faculty of Computer Science at the University of Vienna, Austria. His research interest includes Visualization, Computer Graphics, and Data Science. He received his Ph.D. degree in computer and information science at the Ohio State University, USA. He is a member of the IEEE Computer Society. Contact him at torsten.moeller@univie.ac.at.

Peter Luidolt is the head of the Operation and Maintenance and COO at SOLID Solar Energy Systems GmbH, Graz, Austria. He received his MSc. degree in Physics at the University of Vienna. Contact him at p.luidolt@solid.at.