

Software Requirements Specification

Workflow Diagram Widget



Version 3 - Initialization

Prepared By: TSP

Last Modified: 12 May 2018

Table of Contents

| | |
|--|--------------|
| Workflow Diagram Widget | 1 |
| Version 2 - Initialization | 1 |
| Prepared By: The Solid Project | 1 |
| Last Modified: 12 May 2018 | 1 |
| Introduction | 4 |
| Purpose | 4 |
| Document Conventions | 4 |
| Intended Audience and Reading Suggestions | 4 |
| Project Scope | 4 |
| Overall Description | 4 |
| Product Perspective | 5 |
| Product Functions | 5 |
| User Classes and Characteristics | 6 |
| Manufacturing | 6 |
| Developers | 6 |
| Operating Environment | 6 |
| User Documentation | 6 |
| Assumptions and Dependencies | 7 |
| External Interface Requirements | 7 |
| User Interfaces | 7-8 |
| System Features | 9 |
| Read JSON | 9 |
| Display a Basic Workflow | 10 |
| Allow Customization to the Workflow | 10-11 |
| Create a Developer: How-to | 11 |

Revision History

| Name | Date | Reason For Changes | Version |
|------|-----------|--------------------|---------|
| TSP | 3/12/2018 | Draft | 1.0 |
| TSP | 4/09/2018 | Draft | 2.0 |
| TSP | 5/12/2018 | Final Version | 3.0 |

1. Introduction

1.1 Purpose

The purpose of this document is to describe the requirements for the Workflow Diagram Widget & Shop Floor Visualization. The scope of this project includes continuing Vortek Solutions code using a provided node sample, but after close consideration we decided to move away from Angular 2 and use HTML, CSS Bootstrap, JSON, JavaScript D3, JQuery Libraries, and Python http.server. The Workflow Widget should be customizable. If time is allotted, after the completion of the Workflow Widget and Shop Floor Visualization, other modules such as the scheduling module will be created.

1.2 Document Conventions

This document assumes general knowledge of GitHub and JS terminology and conventions. Some sections of this document include relations to branches and including version control.

This document uses conventions such as bolding and headers to emphasize points. Whilst not all details require its own section, there should be importance on their mention. This document includes hyperlinks, which enable the digital document to link with relative domains (including test and live environments).

1.3 Intended Audience and Reading Suggestions

This document is intended for developers to get familiar with our product and be able to make any modifications or updates to our product.

1.4 Project Scope

The “Workflow Diagram Widget” and “Shop Floor Visualization” are browser embeddable visualization tools built with the intention of being able to visualize workflow diagrams and their relationships. The Workflow Widget is being built to integrate current practices that managers and employees currently use to see potential bottlenecks and pain points during the manufacturing process. The software is built with the intent to visualize not just workflows but other node based relationships such as parent-child cost charts.

2. Overall Description

2.1 Product Perspective

This product is being developed for Northrop Grumman as a tool that displays an overview of how

their workflow is functioning. This product is standalone built on Northrop Grumman's necessity of having a fully functioning, designed interface to replace their current system, and the use of excel spreadsheets. Based upon completion of the Workflow Widget another web based visualization tool for a floor layout, and scheduling will be in order.

2.2 Product Functions

- Widget will accept information regarding workflow passed as JSON
- Widget will display relationships between each node passed within the workflow
- Widget will dynamically adjust UI depending on values within each node

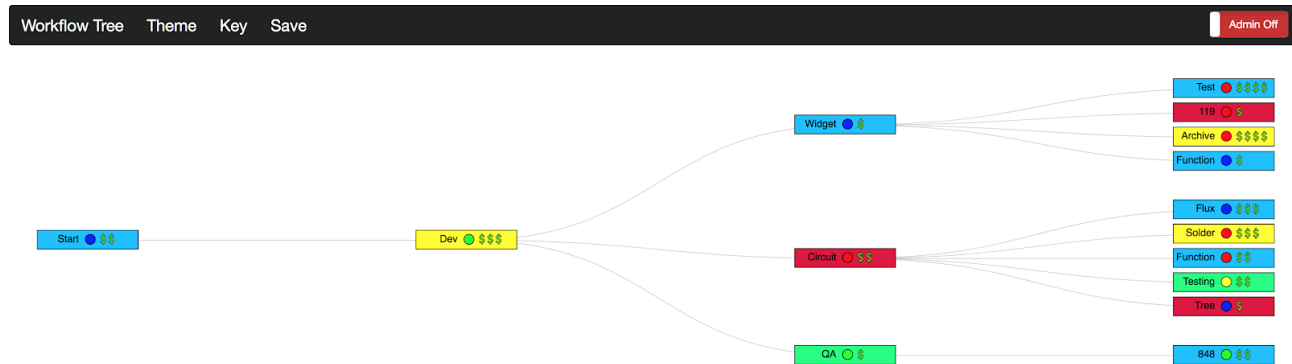
Example of JSON Input:

```

1 {
2   "id": "100",
3   "rev": 0,
4   "iteration": 0,
5   "percentcomplete": 50,
6   "start": "1 / 1 / 2001 13:00",
7   "end": "",
8   "actualhours": 2,
9   "plannedhours": 4,
10  "name": "Start",
11  "desc": "description",
12  "schedule": "S1",
13  "cost": "C2",
14  "quality": "Q3",
15  "children": [
16    {
17      "id": "101",
18      "rev": 0,
19      "iteration": 0,
20      "percentcomplete": 50,
21      "start": "1 / 1 / 2001 13:00",
22      "end": "",
23      "actualhours": 2,
24      "plannedhours": 4,
25      "name": "WIP",
26      "desc": "description",
27      "schedule": "S2",
28      "cost": "C1",
29      "quality": "Q2",
30      "children": [
31        {
32          {
33            "id": "102",
34            "rev": 0,
35            "iteration": 0,
36            "percentcomplete": 50,
37            "start": "1 / 1 / 2001 13:00",
38            "end": "",
39            "actualhours": 2,
40            "plannedhours": 4,
41            "name": "Final State",
42            "desc": "description",
43            "schedule": "S3",
44            "cost": "C3",
45            "quality": "Q4"
46          },
47          {
48            "id": "103",
49            "name": "test",
50            "desc": "description",
51            "schedule": "S4",
52            "cost": "C4",
53            "quality": "Q1"
54          }
55        ]
56      }
57    ]
58  }

```

Result of JSON Input Within Widget:



2.3 User Classes and Characteristics

For use of this product, our goal is to target developers who have the skills to manage and edit CSS documentation, and for those who can manage JavaScript and JSON scripting / object notions. The two main users that will use our product are those in manufacturing and developers.

Manufacturing

Those in manufacturing that will be using this product to determine state history, will allow them to better troubleshoot, and find roadblocks in a workflow. The current system is handled by a series of documents posted on whiteboards and populated reports from different managers.

Developers

Those within the stakeholder group, developers, will use this product and modify the code / themes to match their needs. The goal of this product is to be as customizable as possible so developers can alter the program.

2.4 Operating Environment

This software will run with Internet Explorer 11+, Microsoft Edge, and Firefox 45+

2.5 User Documentation

- D3 API Reference
- User Manual
- GitHub Repository

2.6 Assumptions and Dependencies

This software is dependent on being hosted on a web server in order to execute properly.

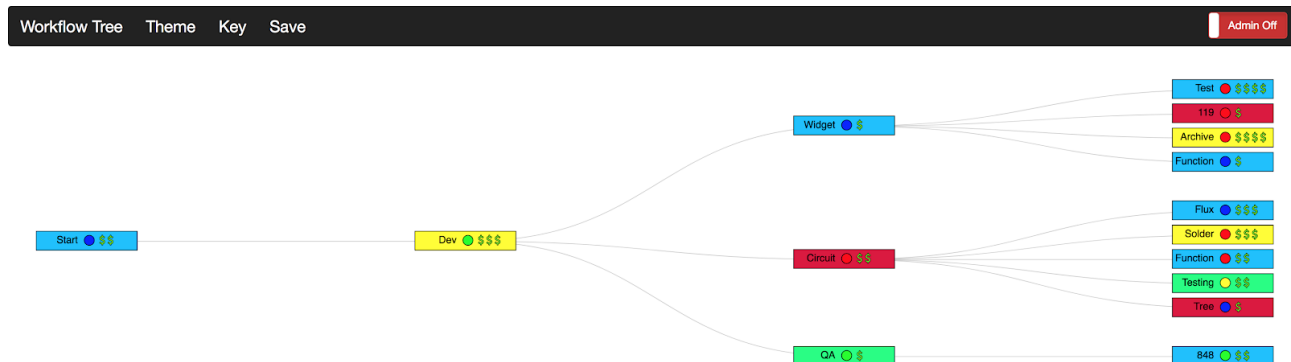
3. External Interface Requirements

3.1 User Interfaces

User interface will follow this **application style guide**:





```
#0072C6 ■ rgb(0,114,198)
#e8e8e8 □ rgb(232,232,232)
#95958d ■ rgb(149,149,141)
#000000 ■ rgb(0,0,0)
#FFFFFF □ rgb(255,255,255)
font-family: "Helvetica Neue", Helvetica, Geneva, Arial, sans-serif;
```

The application UI will adjust based on these passed in conditions:

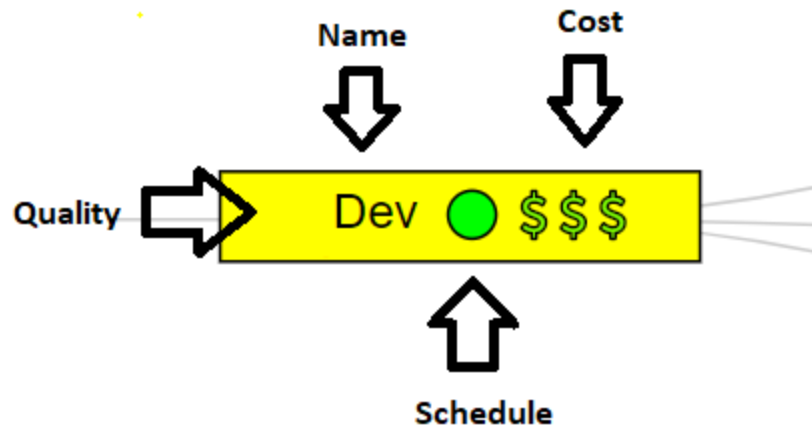


| Cost = Elapsed Hours/Standard Hours | | |
|--|-------------|---------------------------|
| Elapsed hours are less than or equal to standard hours | \$ | Perfect: Cost <= 100% |
| Elapsed hours 1x to 1.5x standard hours | \$ \$ | Good: 100% < Cost <= 150% |
| Elapsed hours 1.5x to 2x standard hours | \$ \$ \$ | Good: 150% < Cost <= 200% |
| Elapsed hours more than 2x standard hours | \$ \$ \$ \$ | Bad: Cost > 200% |

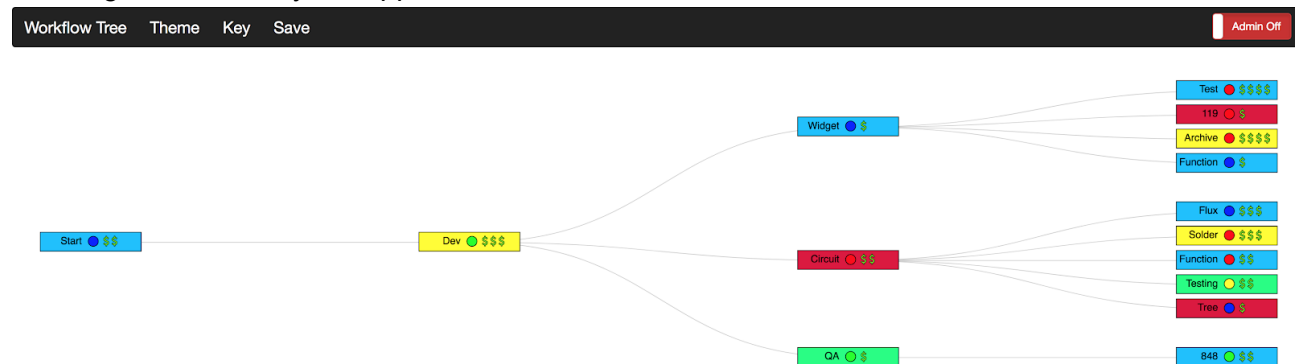
| Quality = WO's with QN/Total WO's | |
|--|---|
| No WO's have a QN against them | Perfect: Quality = 0% |
| None to 25% of WO's have a QN against them | Good: $0\% < \text{Quality} \leq 25\%$ |
| 25% to 50% of WO's have a QN against them | Good: $25\% < \text{Quality} \leq 50\%$ |
| More than 50% of WO's have a QN against them | Bad: Quality > 50% |

| Schedule = Days of QN RW on a WO/Total Open Days of a WO | | |
|--|---|-----------------------------|
| No days of RW operations related to a QN |  | Perfect: Schedule = 0% |
| 0% to 10% of WO open days had RW operations |  | Good: 0% < Schedule <= 10% |
| 10% to 20% of WO open days had RW operations |  | Good: 10% < Schedule <= 20% |
| More than 20% of WO open days had RW operations |  | Bad: Schedule > 20% |

With these adjusted characteristics of each state, they should apply to each node as such:



Each of these nodes will have their relationship passed in with the JSON data and then the resulting tree hierarchy will appear:



4. System Features

Below demonstrates some of the key features this product will offer at completion.

4.1 Read JSON

4.1.1 Description and Priority

In order for our product to work sufficiently, the product must be able to read JSON data and create a workflow using that data.

4.1.2 Stimulus/Response Sequences

The developers involved in this project should be able to input all JSON fields to be displayed and hide all others. This comes with the customizable states, as mentioned later in this section.

4.1.3 Functional Requirements

JavaScript (JS), included in the integrated libraries, will be used to enable the JSON integration. The objects are created and stored in JS.

Using the libraries as a base, JSON object must be configurable to show relevant data to the section that wishes to display it (departments, part locales, etc.).

4.2 Display a Basic Workflow

4.2.1 Description and Priority

In addition to reading JSON, the product must have the ability to display a simple workflow using said data. This is also a top priority.

4.2.2 Stimulus/Response Sequences

The developers involved in this section are the same as mentioned in the JSON section, with the addition of UI / UX developers. Those involved will be charged with reading the JSON input and creating themes to coherently describe their prescribed data.

4.2.3 Functional Requirements

Just like reading the JSON data, the data should be managed by developers or those that are familiar with Object Notation. The difference, however, lies with the stakeholders using the visual aides - they should not need development experience to manipulate the created charts / diagrams.

When an object in the diagram is selected by a user, the next series of steps should appear. This is to minimize screen space and clutter until it is required.

4.3 Allow Customization to the Workflow

4.3.1 Description and Priority

While this step is also critical to the project's success, we must first focus on creating a working product with at least one (1) built-in theme. This task is to be set directly after said success.

4.3.2 Stimulus/Response Sequences

In order to work effectively, the user should be able to create their own style documentation and implement it by altering just a few lines of code.

Upon editing, the user should refer to the documentation How-To, described in the next section, for alterations of this product.

4.3.3 Functional Requirements

With simple CSS manipulation, developers should have the ability to manage / edit endless themes to work with the widget. With simple customization, the widget should still fully function as expected.

4.4 Create a Developer: How-To

4.4.1 Description and Priority

In order for our product to work sufficiently, the product must be able to read JSON data and create a workflow using solely that data. This is the utmost top priority.

4.4.2 Stimulus/Response Sequences

The developers involved in this project should be able to input all JSON fields they wish to be displayed and hide all others. This comes with the customizable states, as mentioned later in this section.

4.4.3 Functional Requirements

JavaScript (JS), included in the integrated libraries, will be used to enable the JSON integration. The objects are created and stored in JS.

Using the libraries as a base, JSON object must be configurable to show relevant data to the section that wishes to display it (departments, part locales, etc.).

Shop Floor Visualization

5. Overall Description

5.1 Product Perspective

This product is being developed for Northrop Grumman as a tool that displays an overview of how their shop floor is functioning. This product is standalone built on Northrop Grumman's necessity of having a fully functioning, designed interface to add to their current system.

5.2 Product Functions

- Visualization will accept information regarding workflow passed as JSON
- Visualization will allow nodes to be dynamic.
- Visualization will dynamically adjust size depending on values within each node.

Example of JSON Input:

```

1 {
2   "id": "100",
3   "rev": 0,
4   "iteration": 0,
5   "percentcomplete": 50,
6   "start": "1 / 1 / 2001 13:00",
7   "end": "",
8   "actualhours": 2,
9   "plannedhours": 4,
10  "name": "Start",
11  "desc": "description",
12  "schedule": "S1",
13  "cost": "C2",
14  "quality": "Q3",
15  "children": [
16    {
17      "id": "101",
18      "rev": 0,
19      "iteration": 0,
20      "percentcomplete": 50,
21      "start": "1 / 1 / 2001 13:00",
22      "end": "",
23      "actualhours": 2,
24      "plannedhours": 4,
25      "name": "WIP",
26      "desc": "description",
27      "schedule": "S2",
28      "cost": "C1",
29      "quality": "Q2",
30      "children": [
31        {
32          "id": "102",
33          "rev": 0,
34          "iteration": 0,
35          "percentcomplete": 50,
36          "start": "1 / 1 / 2001 13:00",
37          "end": "",
38          "actualhours": 2,
39          "plannedhours": 4,
40          "name": "Final State",
41          "desc": "description",
42          "schedule": "S3",
43          "cost": "C3",
44          "quality": "Q4"
45        },
46        {
47          "id": "103",
48          "name": "test",
49          "desc": "description",
50          "schedule": "S4",
51          "cost": "C4",
52          "quality": "Q1"
53        }
54      ]
55    }
56  ]
57 }
58 }

```

Result of JSON Input Within Widget:



5.3 User Classes and Characteristics

For use of this product, our goal is to target developers who have the skills to manage and edit CSS documentation, and for those who can manage JavaScript and JSON scripting / object notions. The two main users that will use our product are those in manufacturing and developers.

Management

Those in management will be using this product to evaluate and design the shop floor. determine state history, will allow them to better troubleshoot, and find roadblocks in a workflow. The current system is handled by a series of documents posted on whiteboards and populated reports from different managers.

Developers

Those within the stakeholder group, developers, will use this product and modify the code / themes to match their needs. The goal of this product is to be as customizable as possible so developers can alter the program.

5.4 Operating Environment

This software will run with Internet Explorer 11+, Microsoft Edge, and Firefox 45+.

5.5 User Documentation

- D3 API Reference
- User Manual
- GitHub Repository

5.6 Assumptions and Dependencies

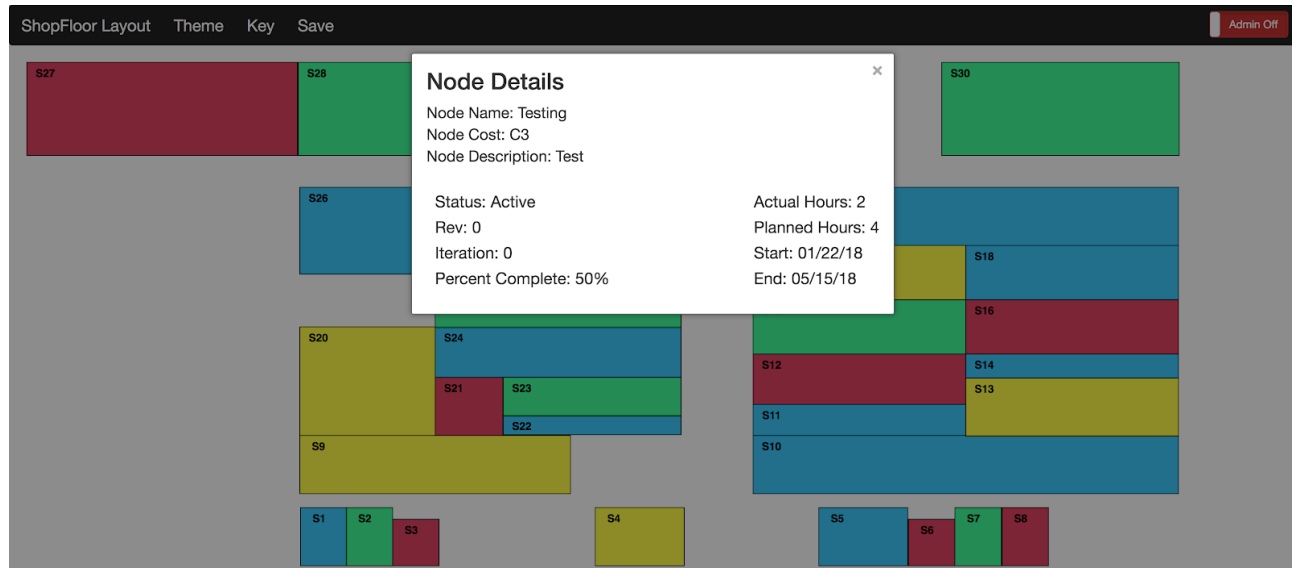
This software is dependent on a framework such as Python or WAMP.

6. External Interface Requirements

6.1 User Interfaces

Each of these nodes will use JSON data and then the shop floor visualization will appear:





7. System Features

Below demonstrates some of the key features this product will offer at completion.

7.1 Read JSON

7.1.1 Description and Priority

In order for our product to work sufficiently, the product must be able to read JSON data and create a workflow using that data.

7.1.2 Stimulus/Response Sequences

The developers involved in this project should be able to input all JSON fields to be displayed and hide all others. This comes with the customizable states, as mentioned later in this section.

7.1.3 Functional Requirements

JavaScript (JS), included in the integrated libraries, will be used to enable the JSON integration. The objects are created and stored in JS.

Using the libraries as a base, JSON object must be configurable to show relevant data to the section that wishes to display it (departments, part locales, etc.).

7.2 Display a Simple Shop Floor

7.2.1 Description and Priority

In addition to reading JSON, the product must have the ability to display a simple shop floor

using said data. This is also a top priority.

7.2.2 Stimulus/Response Sequences

The developers involved in this section are the same as mentioned in the JSON section, with the addition of UI / UX developers. Those involved will be charged with reading the JSON input and creating themes to coherently describe their prescribed data.

7.2.3 Functional Requirements

Just like reading the JSON data, the data should be managed by developers or those that are familiar with Object Notation. The difference, however, lies with the stakeholders using the visual aides - they should not need development experience to manipulate the created charts / diagrams.

When an object in the diagram is selected by a user, the next series of steps should appear. This is to minimize screen space and clutter until it is required.

7.3 Create a Developer: How-To

7.3.1 Description and Priority

In order for our product to work sufficiently, the product must be able to read JSON data and create a shop floor using solely that data. This is the utmost top priority.

7.3.2 Stimulus/Response Sequences

The developers involved in this project should be able to input all JSON fields they wish to be displayed and hide all others. This comes with the customizable states, as mentioned later in this section.

7.3.3 Functional Requirements

JavaScript (JS), included in the integrated libraries, will be used to enable the JSON integration. The objects are created and stored in JS.

Using the libraries as a base, JSON object must be configurable to show relevant data to the section that wishes to display it (departments, part locales, etc.).

Original document template copyright:

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.