****

**Component Level Model Based Assessment Tool Development**

**Metal Whiskers Simulation**

**June 21, 2023**

Table of Contents

[Introduction 3](#_Toc141110561)

[Modules Overview 3](#_Toc141110562)

[Visio® Logic Diagram Descriptors – Module 1 4](#_Toc141110563)

[Visio® Logic Diagram Descriptors– Module 2 7](#_Toc141110564)

[Module 3 8](#_Toc141110565)

[Limitations of Unity® Game Engine and Recommendations 9](#_Toc141110566)

[Future Possibilities and Guidance 9](#_Toc141110567)

[References and Information 9](#_Toc141110568)

[Figures 11](#_Toc141110569)

# Introduction

For decades, metal whisker induced failures have been a significant problem. Multiple organizations would like to use advanced modeling, in conjunction with Monte Carlo simulation techniques to assess the probability of detached metal whiskers falling on a circuit card assembly (CCA) within an electronic enclosure, causing a conductive bridge that may result in a system failure.

# Modules Overview

This paper provides the initial system documentation to codify goals of an on-going mission assurance assessment tool capstone project to simulate tin whisker damage potential in hardware components within electronic enclosures for risk evaluation purposes. The tool requirements are delineated using Visio®-generated logic diagrams, shown in Figures 1 through 9, to address key functional system modules of the tool. As such, three key individual modules are described as follows:

Module 1: Compute tin whisker probability of mechanical bridging (contact with electronic node pairs).

Module 2: Evaluate the impact of the bridging with respect to the electronic behavior of the circuit board.

Module 3: Evaluate the consequence of the output of Module 2 and the severity associated with a potential failure of electrical components (failure of function or degradation of capability).

Modules 1 and 2 are intended to be for 'general purpose' application and are being developed using public domain open source software versions of script and tool builds to prove capability and functionality. Module 3 is expected to be developed through user specific controlled versions of software satisfying security issues due to the specific application to evaluate the risk in terms of system consequence and a foreseen need to import proprietary computer-aided design files of CCAs.

**Key:**

AU Team 1 Completed

AU Team 1 Partially completed/ Expanded Line Items

AU Team 2 Completed

AU Team 2 Partially completed/ Expanded Line Items

# Visio® Logic Diagram Descriptors – Module 1

Figure 1: Inputs

1. Location: The location of the inputs and outputs will be specified.
2. Units: The user will have the option to enter the desired units for both the inputs and outputs.
3. Tin whisker definition: Input data specifying the characteristics of the tin whiskers.
4. Distribution type: The statistical distribution that will be used for modeling, such as a log-normal distribution.
5. Physical characteristics: User input to change the characteristics of the tin whiskers being used.
6. Number: Number of whiskers being dropped
7. Size: Size of whiskers
8. Material: Material of the whiskers, such as tin, zinc, etc.
9. Source location: Where the tin whiskers may be generated from, such as the thin air, or physical components. The location on the circuit board that the whiskers will be spawned from is also specified.
10. Physical limits: The maximum physical limitations of tin whiskers to represent the expected growth.
11. Initial condition: The location of the whiskers at the beginning of the simulation. Such as:
12. Free-floating: Not attached to any objects; In the air without constraints
13. Attached: Attached to the metal surface that they were grown from
14. Mixed: A mixture of both free-floating and attached whiskers. The user may define the percentage of free-floating and attached whiskers created.
15. Physical component definition: The physical components and bounds that the whiskers will be subjected to.
16. CCA file: A file that may be uploaded to represent a printed circuit board.
17. Material: The materials of the electrical components being represented on the circuit board must be specified within the CCA file prior to being imported.
18. System container/Bounding box: The physical space designed to constrain the simulation.
19. Physical environment: The physical conditions that will affect the placement of tin whiskers, including:
20. Shock
21. Vibration
22. Orientation
23. Acceleration: Including gravity, and the rate at which the whiskers fall (constant velocity, accelerating, etc.)
24. Specify output data
25. Volume of output data desired: How many shorts are outputted based on the users selection.
26. Text file: A separate text file of the outputs available to the user.
27. Upload previous input file: The ability for the user to upload a previous file that consists of the inputs they used.

Figure 2: System Level Function

1. Identify and label node pairs
2. Assign pad numbers to exposed surfaces: In order to identify shorts, each exposed pad must have a unique ID number assigned to it. Refer to physical component definition in Figure 1-15.
3. Geometric surface: Description of the physical surface of the node, such as the shape.
4. Coordinates: The x, y, z coordinates of the pad must be attached to the ID number in order to identify it.
5. Label ID pad number: The ID number must be labeled at the node in order to keep track of the location.
6. Graphic display: The graphical user interface available to the user in order to identify the node pairs.
7. Geometry: The physical geometry of the pad.
8. Pad number and location: The pad ID number appears at the correct location of the pad.
9. Whisker generator: How the whiskers will be present in the simulation.
10. Create ID number for each whisker generated: Unique number assigned to each whisker.
11. Define initial starting position of whiskers: Location where whisker will begin. The location will be determined by the random number generator, Figure 2-14.
12. Bounding box: The location of the whiskers must remain within the bounding box from Figure 1-18.
13. Limits of component: Each whisker must not exceed the specified length, width, and thickness specified as an input. Neglect components less than the minimum distance between two nodes. If the dimensions of the whisker are less than the minimum distance between two nodes, these whiskers will not be generated.
14. Random number generator: The numbers generated will result in the location of the whiskers, Figure 2-11.
15. Distribution: The numbers generated will align with the distribution type defined in Figure 1-4.
16. Maximum and minimum range: The limits on the coordinates that may be generated.
17. Bounding box coordinates: The coordinates of the bounding box that all whiskers must remain within.
18. Coordinates of pad: The whiskers may not be generated at a location that a pad already exists.
19. Tag database: A database that stores the following:
20. Whisker ID number and location: From Figure 2-10.
21. Node pair ID number and location: Data from identifying and labeling node pairs in Figure 2-2.

Figure 3: Simulation

1. Identification database: The data collected in Figure 2-19 that will be used for the simulation itself.
2. Physical environment history: From Figure 1-19.
3. Tin whisker initial location: From Figure 2-11.
4. Circuit and physical enclosure: From Figure 1-15.
5. Tag database: From Figure 2-19.
6. Identify where whisker touches pad: Using Figure 3-7 and Figure 3-8, any coordinates that a whisker and pad touches will be identified.
7. Whisker endpoint location: The endpoint coordinates of the whisker, extracted from the identification database.
8. Coordinate of pad: The pad coordinates extracted from the identification database.
9. Identify if nodes were bridged by whiskers: Using the identified whiskers and pads identified in Figure 3-6, two pads in contact with the same whisker will be identified as bridged.
10. Store whisker ID and pad ID of bridged nodes: The identified bridged nodes ID and whisker ID will be stored.
11. Identify surface geometry of node where whisker touches the pad: What the bridged pad geometry looks like, as extracted from the identification database from Figure 3-1.
12. Contact database: The whisker and pad ID of bridged nodes will be stored in a database. Each contact will be identified as one of the following
13. Whisker in contact with whisker: Any whisker in contact with another whisker must identify the additional whisker.
14. Whisker in contact with pad: Any whisker that is touching a pad.
15. Location of contact: Coordinate where the contact occurred.
16. Short database: Identified the shorts generated within the simulation by the whiskers.
17. Whisker connected pads: Where the short occurs
18. Number of whiskers connected in a “chain” that results in a short: How many whiskers were used to create a short.
19. Whiskers between pads, creating a short: The ID of each whisker between two pads that create a short.

Figure 4: Outputs

1. Short circuit: From the short database in Figure 3-16, the components and whiskers resulting in a short circuit will be identified.
2. ID of whiskers resulting in short: Output what whiskers resulted in a short.
3. Number of shorts: Output how many shorts were created.
4. Location of pads resulting in a short: Output of the location of the pads, including the pad ID.
5. Storable results file: Ability for user to save the results of the simulation, as well as the inputs inserted.
6. Databases: Save the identification database, contact database, and short database.
7. Inputs: Store the initial input parameters.

Figure 5: Graphical User Interface (GUI)

1. Inputs: For user to input values to run the simulation.
2. Perspective: Ability for user to move around the circuit, and view from different angles.
3. Screen capture: Ability to take screen shots or recordings of the circuit, including when the tin whiskers may be moving around.
4. Display whiskers and PCB: 3D model of the circuit board and whiskers that are easily identifiable.
5. Highlight shorted whiskers and pads: Distinction of shorted whiskers and pads that are easily identifiable for the user visually.
6. Report of results: For user to view and analyze the results of the simulation.
7. Store on device: Ability for user to easily store the result file from Figure 3-16.
8. Store databases: Store databases in location from Figure 1-1.
9. Edit circuit: Add components, delete components, and modify the original circuit.
10. Pad identification: Allows user to view and modify the pad identifications from the identification database.

# Visio® Logic Diagram Descriptors– Module 2

Figure 6: Inputs

1. Read in existing file: Allows the user to upload any previously existing files to be evaluated.
2. Component tolerance range: The maximum and minimum voltage that an electrical component is capable of operating within. This will help increase the accuracy of the simulation.
3. Circuit analysis (Module 1): Imports the data inputted and outputted in Module 1.
4. Electronic circuit diagram: This is the physical layout of the electrical circuit, separate from the CCA file.
5. GUI: The graphical user interface presented to the user, expanded on from Module 1 to include the electronic circuit diagram.
6. **Assign ID to location on electronic circuit diagram\***: Must be done by user, eventually can be done by AI.

\*It is extremely important to verify it is tagged correctly, otherwise false conclusions may be made

Figure 7: Circuit Functionality

1. Short database: Extracted from Module 1. Includes all of the whiskers, pads, and locations of shorts in the circuit. See Figure 4.
2. Normal: Circuit behaves as expected.
3. Abnormal: There is a problem in the circuit that is preventing it from behaving as expected.
4. Error: If abnormal behavior is detected, an error will be presented to the user.

Figure 8: Program Processing

1. Standard: How the circuit is expected to behave.
2. Individual shorts: A singular short in the circuit.
3. Compare individual to standard: Allows user to evaluate the effect of individual shorts in comparison to the standard circuit.
4. Collective shorts: Multiple shorts in the circuit.
5. Compare collective to standard: Allows user to evaluate the effect of a specified number of collective shorts in comparison to the standard circuit.

Figure 9: Outputs

1. Recommend validation test: If the circuit does not behave as expected, a physical validation test will be recommended.
2. Save interim files: The current data inputted and outputted may be saved at any moment, and retrieved at another time. A time stamp will be associated to each interim file to prevent overwriting previous files.
3. Output to file: The user has the ability to download the results of the simulation, and the inputs will be stored.

# Module 3

A system engineer with a satisfactory understanding of the complexities of the system as a whole must evaluate the consequence of the output of Module 2 and the severity associated with a potential failure of electrical components. In order to make an accurate assessment, the inputs from Module 1 is also required for the system engineer. This process cannot be automated, as it requires human discretion. The systems engineer will use the probability of an event in order to evaluate the potential consequences, such as: loss of mission, loss of software functionality, or degradation of capability.

# Limitations of Unity® Game Engine and Recommendations

* Objects may morph through other models.
* Considerable size gap between circuit board and whiskers.
* Measurement units are not specific, need to be defined.
  + Need to define English and metric units within Unity® to remain consistent and prevent errors.
* Lag issues due to the high volume of the simulation.
* Time steps may limit the small details of the simulation.
* To verify the tool produces realistic results, it is necessary to run a trial to ensure the expected physical results are consistent with the calculated results from the simulation.
  + Pre-defined inputs to verify this trial, especially when the tool is modified. These results will be compared to the calculated results.

# Future Possibilities and Guidance

* Once this tool has been developed, the capabilities of the tool may be expanded beyond just metal whisker evaluations. Foreign Object Debris (FOD) may also be simulated with this tool in order to evaluate risk.
* The environmental capabilities may be expanded to other situations not represented in the initial requirements, as the tool is developed and matured.
* Human input may be easily expanded to AI capabilities for analysis.
* Proprietary circuit analysis may be used with this tool once the appropriate security measures are in place.
* The integration of Module 2 output into Module 3 may be more clearly defined, such that the system engineer understands the limitations and intentions of the results generated.
* As blind spots are identified, the tool must be updated, and the supporting documentation of the limitations must be shared with others.
* Aerodynamic study of metal whisker in different air densities may increase the accuracy of this tool.

# References and Information

For more information regarding tin whiskers, visit the NASA website, <https://nepp.nasa.gov/whisker/>.

For more information on the Unity ® game engine, visit the Unity® website, <https://unity.com/open-projects>.

Preferred programming languages are C#, C++, and Python®. Software may be translated between languages. Software is being developed for a Windows compatible environment. All source code must be identified and tagged for others use, as well as a time-stamp.

# Figures



Figure 1



Figure 2



Figure 3



Figure 4



Figure 5



Figure 6



Figure 7



Figure 8



Figure 9