World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:17, No:1, 2023

Analyzing the Relationship between the Systems Decisions Process and Artificial Intelligence: A Machine Vision Case Study

Mitchell J. McHugh, John J. Case

***Abstract*—**Systems engineering is a holistic discipline that seeks to organize and optimize complex, interdisciplinary systems. With the growth of artificial intelligence, systems engineers must face the challenge of leveraging artificial intelligence systems to solve complex problems. This paper analyzes the integration of systems engineering and artificial intelligence and discusses how artificial intelligence systems embody the systems decision process (SDP). The SDP is a four-stage problem-solving framework that outlines how systems engineers can design and implement solutions using value-focused thinking. This paper argues that artificial intelligence models can replicate the SDP, thus validating its flexible, value-focused foundation. The authors demonstrate this by developing a machine vision mobile application that can classify weapons to augment the decision-making role of an Army subject matter expert. This practical application was an end-to-end design challenge that highlights how artificial intelligence systems embody systems engineering principles. The impact of this research demonstrates that the SDP is a dynamic tool that systems engineers should leverage when incorporating artificial intelligence within the systems that they develop.

***Keywords*—**Computer vision, machine learning, mobile application, systems engineering, systems decision process.

I. Introduction

THE focus of this study was to conduct an end-to-end application of systems engineering thinking and artificial intelligence (AI) to solve an engineering challenge. The contribution from this study is the analysis of the relationship between AI and systems engineering. Ultimately, this study will attempt to expand the use of systems engineering design principles in the development of AI and encourage future systems engineers to utilize AI techniques when solving engineering challenges.

The expansion of AI systems is revolutionizing the modern world. AI and machine learning models solve problems and provide insights never thought possible. Because of the utility and potential of AI techniques, systems engineers must understand how to effectively leverage them. While the SDP is a flexible and universal process, the systems engineering community has yet to thoroughly discuss its implementation with AI. This study is significant because it will facilitate discussion on the integration of AI and systems engineering.

We will demonstrate the interrelationships between AI and systems engineering in a real-world scenario: In a typical army unit, there are several experienced non-commissioned officers (NCOs) who are experts at their jobs and know every detail relating to training, equipment, or personnel. Junior unit members, however, do not share the same level of expertise and therefore require guidance from the senior NCOs. The practical application of this study creates an AI tool to fill the role of the senior NCO for a specific task, therefore freeing the NCO to address other serious issues. The scenario used to demonstrate this problem is the task of identifying equipment and its common maintenance requirements.

This study represents a balance of holistic and technical skill. The tangible product of this study is an end-to-end machine vision mobile application built with technical skills like coding, computer science, and mathematics. However, the significant contribution of this study is the exposed potential of the combination of the SDP's approach to systems thinking and AI.

II. Background

The SDP is a four-stage process that focuses on general problem-solving principles and value-focused thinking. Fig. 1 visually represents the four stages of the SDP and its nested tasks.

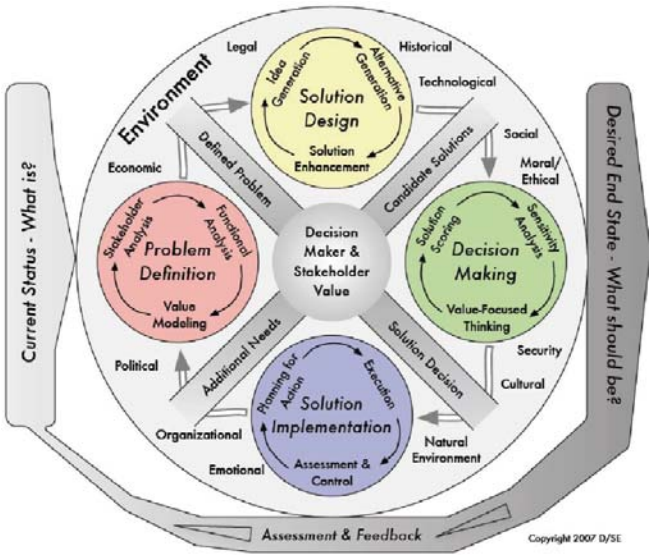

Fig. 1 The SDP [1]

Mitchell McHugh is with Department of Systems Engineering, United States Military Academy, West Point, NY 10996, USA (corresponding author, e-mail: mitchell.mchugh@westpoint.edu).

Major John Case is with Department of Systems Engineering, United States Military Academy, West Point, NY 10996, USA (e-mail: john.case@westpoint.edu).

The SDP is useful because it is versatile in its application. Therefore, the SDP will become exceedingly relevant as the combination of systems adds complexity that cannot be simplified by engineers creating more specific engineering processes [1].

The SDP beings with a problem definition phase in which stakeholder analysis, value modeling, and requirement analysis occurs [1]. During the problem definition phase of SDP, the systems engineer attempts to fully understand the context and background of the problem, those individuals and organizations with a stake in the problem, and requirements that they must meet to solve the problem.

The second phase of the SDP, the solution design phase, consists of idea generation, cost analysis, and alternative generation [1]. During this phase, the systems engineer will develop a variety of candidate solutions to the problem while also developing a method to assess cost. At the completion of the solution design phase, the systems engineer will have several potential solutions that they will be evaluate based on value and cost.

The evaluation of each candidate solution takes place during the decision-making phase of the SDP. For each candidate solution, the systems engineer generates a trade space between value and cost and conducts sensitivity and tradeoff analysis [1]. The systems engineer then selects the best solution based on the value-cost tradeoff while also remaining cognizant of the output of the sensitivity and tradeoff analysis. Lastly, the systems engineer develops and improves the selected solution before the solution is executed.

Planning, execution, and monitoring and controlling of the solution all take place within the solution implementation phase of the SDP [1]. After the decision maker decided upon a solution, the planning begins to ensure the effective execution of the solution. Oversight through monitoring and controlling are critical throughout planning and execution actions to ensure the solution is implemented in a manner consistent with stakeholder needs.

Despite the SDP's proven flexibility and versatility, the systems community has not yet analyzed its relationship with AI systems. Petrotta and Peterson studied the potential benefits of and an early framework for augmenting human intelligence with AI systems, but this field of study remains in its early stages [2].

III. Methodology

A. The Question of Integration

The AI field is rapidly growing and continuing to solve complex problems. As the world and its systems become more complex and interdependent, systems engineers will need to learn how to leverage AI techniques to solve problems and support their informed decision making. Despite the need for the integration of AI and systems engineering, there have been problems with applying AI models to systems engineering, specifically software systems engineering [3]. Therefore, the question at hand is: how well are the SDP and AI models postured to integrate?

At first glance, one may look to see where an AI model fits into the existing structure of the SDP. However, the integration of AI and systems engineering is much more complex than treating an AI model as a single component of the SDP. To answer this question, it is helpful to go back to the roots of the problem and realize what we are truly trying to integrate. As defined by Petrotta and Peterson, AI is the "theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages" [2]. Considering this definition, the connection is made between the SDP and AI. AI models do not fit into the SDP as a supplemental tool, rather, AI models fulfill the SDP because they can perform the human-intelligent task of decision-making. If AI models are well designed, fully replicating a decision-making process, then the AI model fulfills the SDP.

B. AI and the SDP

The SDP begins with a problem definition phase in which the systems engineer grasps the scope of the problem and truly understands what factors they need to consider while decision making. An AI model performs problem definition in its inherent design. The first iteration of an AI model's problem definition requires a software engineer to build the model in accordance with stakeholder analysis and the initial problem definition. However, after the model is trained, its design encapsulates the problem definition phase. In the example of a supervised classification AI system, the trained model, given an unlabeled piece of data, analyzes that piece of data with using model's weights developed in the training process to be able to make an informed decision. Similarly, a systems engineer analyzes information related to a problem with respect to stakeholder value or weights in order to make the most informed decision possible.

The second phase of the SDP is the solution design phase, in which a systems engineer generates potential solutions or alternatives to the problem. When creating a solution space, there is a key distinction between AI models and systems engineers. AI models are inductive learners, they learn from specific historical examples and generalize based on those specific examples. On the other hand, a systems engineer generally creates solution alternatives through deductive inference, essentially creating specific solutions based on general, holistic premises. For example, a systems engineer's logic would argue: "All alternatives that meet these requirements are feasible. These three alternatives meet the requirements. Therefore, the three alternatives are feasible." An example of an AI system's inductive learning would say "All alternatives with this attribute have been deemed feasible. Alternative 1 has this attribute. Alternative 1 is likely feasible." Regardless of the difference in learning reasoning used to create a solution space, an AI system creates a solution space, nonetheless.

In the third phase of the SDP, a systems engineer chooses the best alternative available using value-focused thinking. AI models also conduct a decision-making phase in which they use

value to make a decision, or prediction. AI models commonly define value as accuracy. A predictive AI model assigns the most value to an alternative with the highest probability, based on inductive learning from data. Regardless of the metric, AI models fundamentally use value to make a decision, just as a systems engineer does.

In the last phase of the SDP, solution implementation, a systems engineer plans, executes, and controls their chosen alternative. The solution implementation phase for an AI model is limitless in its opportunities. An AI model may be a part of a larger AI system. An AI model is the computer program that processes the data and performs the mathematics and prediction generation. The AI system would include sensors that collect the data, the model that performs the algorithms, and software that takes the model output and gives direction to hardware to perform an action. An example of this would be the AI system in a self-driving car. The self-driving car takes in real-time data from its environment, performs the algorithm to interpret the data, and responds to the algorithm output by applying the brakes, turning the steering wheel, or continuing its path. Additionally, a software engineer can design a feedback loop an AI model for control. Essentially, if the AI model makes an incorrect decision or prediction, the feedback loop will route that prediction back into the model as input to retrain the model for a more accurate decision in the future. AI models are fully capable of implementing, executing, and controlling decisions that it makes, just as a systems engineer does.

An AI model's replication of the SDP is a true validation of the design of the SDP, given that the SDP's genesis preceded the AI revolution. The SDP's flexibility and holistic framework makes it applicable to the AI field. The SDP can be a powerful roadmap for AI engineers to design AI models and AI systems to meet stakeholder requirements. An AI model designed using the SDP would effectively capture the systems engineering process, facilitating more efficient and effective AI models.

IV. Practical Application of AI and Systems Engineering

A. Machine Vision Background

This study utilizes a machine vision case study to show the utility of the SDP in designing an AI system. Machine vision AI systems replicate human vision tasks like image classification. One of the utilities of machine vision AI systems is the ability to complete the complex task of image classification. A software developer may adjust a machine vision AI system in numerous ways to fit the need of the problem they are attempting to solve. Significantly, a software developer can optimize machine vision AI models by manipulating data, utilizing transfer learning, and model compression to allow the model to be deployable on a variety of devices.

Because of the importance of data quality and quantity, the data collection stage of building an AI model is arguably the most important and most time-consuming stage. Image data augmentation is an especially important data acquisition technique with respect to machine vision. Image augmentation can be very helpful when image data are not readily available. Image augmentation is essentially the transformation of image data to enrich the dataset and prevent model overfitting by integrating controlled variance into a dataset. An image recognition model will perform better if the software developer uses data with built-in variance in lighting, color, orientation, etc. [4].

Transfer learning is a process in which a developer uses the base network of a pre-trained model and only conducts training on the top layers of the model. Transfer learning takes advantage of the pre-trained model's ability to recognize higher level features and objects while allowing the developer to fine tune the model to the intended research task by training only the top few layers [5]. There are many state-of-the-art pre-trained models that developers have open-sourced and are available for software engineers to leverage with transfer learning. One of these models, MobileNet, is particularly well-suited for machine vision mobile applications [5].

Utilizing transfer learning and MobileNet architecture, a developer can train a custom image classification model, but must find a way to convert their model to a mobile-friendly version. This can be done by using model optimization tools, like TensorFlow Lite, which performs several optimizing functions to accelerate model speed, efficiency, size, and accuracy for mobile use [5]. In addition to model optimization, software engineers may use model compression techniques to further reduce model size and allow for mobile deployment [6]. Model compression is an effective way to reduce model size for mobile deployment while improving model performance.

B. Practical Application Scenario

To demonstrate AI's validation of the SDP, we performed a practical demonstration. The application of conclusions made regarding the integration of AI and systems engineering will guide the methodology used to address the scenario of limited equipment subject matter expertise in an Army unit. Traditionally, equipment maintenance knowledge and expertise are consolidated within several SMEs (NCOs or Warrant Officers). This is simply a result of knowledge gained from years of experience working with specific Army equipment. New soldiers and junior NCOs, however, do not have the same subject matter expertise would benefit from an AI system. The expertise and leadership of the senior NCO could never be replaced; however, an AI tool would allow the NCO to focus less time on maintenance and more time leading and developing soldiers. The initial concept of this tool was to create a mobile application that could identify Army equipment using a machine vision model and return common part failures and other maintenance data corresponding to that piece of Army equipment.

C. Data Collection and Preparation

Creating an image classification machine vision model began with data collection. To reduce complexity of the initial model design, we selected the M4, M240, and M9 weapon systems to be the primary weapons classes. We created an initial image set for each weapon system by physically taking pictures of the

weapon system from different angles and with different backgrounds. Fig. 2 displays example data from the initial image dataset.

Fig. 2 Initial Image Data Examples: M9, M4, M240, respectively

Using the initial image data set, we performed image augmentation to create a large enough data set to train an effective model. 75 images belonging to the three weapons classes composed the initial data set. Performing the image augmentation yielded 5,986 images for training the model and 2,687 images for validating the model. The objective of the image augmentation was to create a robust data set to properly train and validate the model.

D.Model Training, Validation, and Evaluation

With the full image data set on hand, model building could begin. We leveraged transfer learning in building the model to take advantage of a pretrained model architecture already oriented towards mobile deployment. MobileNetV2 is a mobile architecture with its lower-level layers already trained to perform image classification on the 1000-class ImageNet dataset [7]. Because MovileNetV2's developers designed the architecture to be a compact, efficient image classifier, it was not necessary to retrain the entire model. Adding and training a dense and softmax layer will add weights to the model that are specific to classifying the weapon systems while retaining the lower-level features already present. Fig. 3 highlights the new model summary, a combination of MobileNetV2 and the added layers.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Functi (None, 7, 7, 1280)        2257984
_________________________________________________________________
flatten (Flatten)            (None, 62720)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              64226304
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 3075
=================================================================
Total params: 66,487,363
Trainable params: 64,229,379
Non-trainable params: 2,257,984
_________________________________________________________________
```

Fig. 3 Model Summary

To finish the data preparation process, we divided the 5,985 training images by class, resulting in 1999 M4 images, 1566 M240 images, and 2420 M9 images. The 2,687 validation images were also divided by class, resulting in 1073 M4 images, 716 M40 images, and 894 M9 images. Lastly, we used 17 random weapons images from the internet as test images. Fig. 4 displays images from the test image dataset.

Fig. 4 Test Image Data: M4, M9, M240, respectively

With the training, validation, and test image data prepared, we could proceed to compile and train the model. The model summarized in Fig. 3 was compiled using the Adam optimization algorithm, a categorical cross entropy loss function, and used model accuracy as the metric of interest. Prior to training the model, we implemented early stopping and model checkpoints to avoid unnecessary training after reaching the optimal validation accuracy and to save the optimal model weights to avoid retraining the model in the future. We trained the model for seven epochs with a batch size of 800 images. Fig. 5 displays the Training and Validation Accuracy and Loss.

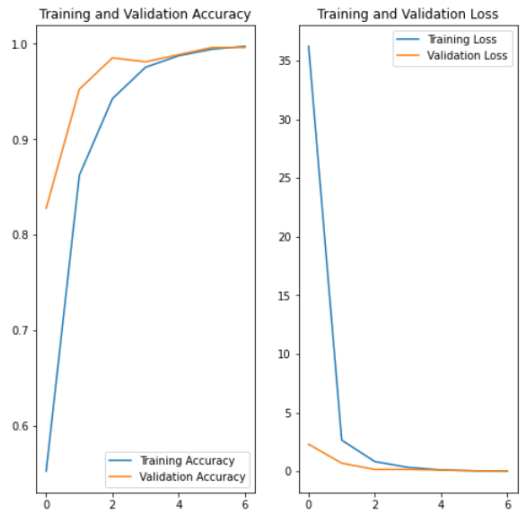

Fig. 5 Training and Validation Accuracy and Loss Plots

We evaluated the model using the test data and returned a loss of 82.34%. The visual representation of the evaluation process displays the test images with their respective classifications in a matrix. A "1" in indicates the prediction with each column corresponding to an image class: M240, M4, and

M9, respectively. Fig. 6 displays that visual representation of the evaluation process.

Fig. 6 Visual Representation of Evaluation Process with Classification Matrix

Looking at the classification matrix, the model incorrectly classified the first image, indicated by the "1" in the M240 column. This incorrect prediction is likely due to the soldier in the image. The training image data did not include any soldiers holding the M4, therefore the model did not learn how to classify the M4 with the addition of the soldier. We could avoid similar incorrect predictions in the future by including images of soldiers holding the M4 in the training data. However, the model correctly classified images two, three, four and five, as indicated by the "1"s placed in the correct column in the classification matrix. This prediction result follows the 82% test accuracy. The test images dataset included images from the internet that were not in the training set to reflect the practical application of the model. When soldiers use the application, there will likely be different attachments on the weapon systems and the backgrounds of the images may not be a solid color. Using different test images that included these variations validated that the model could recognize the weapon system regardless of the background and potential modifications to the weapon system.

This prediction model file was 253MB and too large to use on a mobile device. In order to be able to use the model file on a mobile app, we compressed the model using post-training quantization. Post-training quantization converts a full-size model into a compressed file while also improving model efficiency with little effect on model accuracy. We used dynamic range quantization to convert and compress the model because of its relative simplicity and high compression capability. After converting and compressing the model, its file size was 18MB, a 93% reduction in size. With the significant reduction in size, the model was ready for mobile deployment.

D. Mobile Application Development

Development of the mobile application began with leveraging an open-source application framework developed by TensorFlow [8]. The TensorFlow application framework already included code and the structure for a quantized classification model and therefore, all that was required was to replace the default quantized model with the custom quantized project model and adjust the supporting code to account for the class changes. After changing the application aesthetics, the application successfully runs on mobile Android devices and was able to classify the weapons of interest using the device's camera. We tested the application on the Samsung Galaxy S8 Active running the Android 9.0 operating system. Fig. 7 displays a screenshot of the application running on the external mobile device while actively classifying a weapon system. The application also displays the probability of each class prediction given the image rendered by the device camera. This is a useful feature because it captures the confidence of the model's prediction.

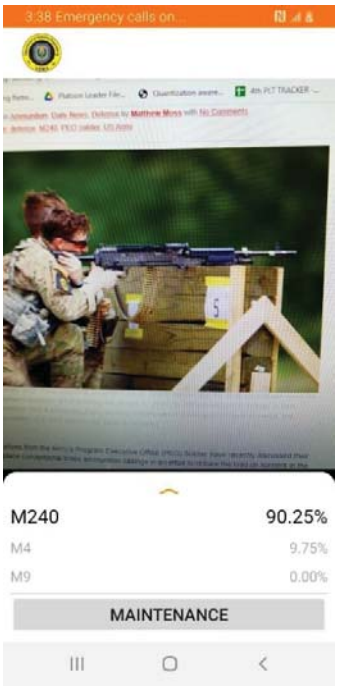

Fig. 7 Weapons Classifier Application Running on Mobile Device

V. Conclusion and Recommendation for Future Research

A. Practical Application Conclusions and Recommendations

The successful development of an image classification mobile application was a practical example of how AI systems validate and replicate the SDP. The design of the application and model training represented the problem definition phase of the SDP. Knowing that the stakeholders for the model would primarily be junior soldiers, we designed the model to be deployed on an easy-to-use mobile device application. During model training, the model generated weights using image data that the model would use later when classifying new objects. Essentially, the model analyzed weapon image data and learned what parts of the image data it needed to consider when making a classification. This process is similar to how a systems engineer conducts research and stakeholder analysis to understand what they need to consider when making a decision.

The development of the different weapons classes and model weights reflected the solution design phase of the SDP. Following the inductive learning technique that AI systems use, the model creates a solution space based on its learning from the labelled weapons images. The AI model's weights represent specific attributes that correspond to a known weapon system. When the model processed a new image to classify, it analyzed the image and looked for specific attributes. Any classification

that had attributes that matched the unknown weapon would be added to the solution space. The model's logic can be stated as, "Any weapons classes with these specific attributes are potentially the unknown weapon system. I have learned that the M4 and M240 weapons classes have those specific attributes. It is likely that the unknown weapon is an M4 or M240." While a systems engineer would traditionally use deductive reasoning to generate alternatives, the AI model selects among alternatives, nonetheless.

The AI model performs the decision-making phase of the SDP when it calculates accuracy for each of its predictions. A systems engineer uses value-focused thinking when making a decision to select the alternative with the most value. Similarly, the AI model defines value as confidence, or probability, and classifies the unknown weapons system based the highest probability. Fig. 5 displays the probability of each prediction to the right of the classification.

The software engineer can add supplementary functionality to the mobile application to fully complete the solution implementation phase of the SDP. The AI model made a value-focused prediction and presented the weapons classification. That classification could return maintenance data, common faults, or specifications for the weapon system. Finally, the application could include a feedback-loop to constantly re-train the model when it makes an incorrect weapons classification, based on user feedback. Similar to a systems engineer, the AI model can implement its decision depending on the needs of the stakeholder.

B. Future Research

Future research that focuses on expanding the functionality of the weapons classifying mobile application would be helpful in demonstrating the variety of opportunities that come from implementing an AI model's classifications. These features would more fully illuminate the fulfillment of the SDP's solution implementation phase. Additionally, the systems engineers can further refine the SDP to facilitate integration with AI systems by specifying a data collection phase during problem definition. Although the problem definition phase includes background research, the data collection process is so essential to developing a quality AI model, it needs to be enumerated. Additionally, even without the application of an AI model, data collection is still relevant to the SDP, reminding systems engineers to collect enough quality data to make an informed decision.

C. Conclusion

The relationship between systems engineering and AI is becoming increasingly important as the modern world becomes more interdependent and reliant on AI systems. This study has shown that AI systems can effectively replicate the SDP and help systems engineers make informed decisions. AI's inherent ability to replicate the SDP also validates the SDP as an effective decision-making process based on objective reasoning and value-focused thinking. In the future, systems engineers should consider how to effectively leverage AI systems to supplement their problem-solving processes. This will result in more intelligent systems that can benefit from the value provided by AI.

REFERENCES

[1] Parnell, G. S., Driscoll, P. J., & Henderson, D. H. (2011). Decision Making in Systems Engineering and Management. Hoboken: John Wiley & Sons, Inc..

[2] Petrotta, M., Sterling Heights, M. I., & Peterson, T. (2019, July). Implementing Augmented Intelligence in Systems Engineering. In *INCOSE International Symposium* (Vol. 29, No. 1, pp. 543-543).

[3] Sommerville, I. (2002). *Artificial Intelligence and Systems Engineering.* Lancaster University.

[4] Sessions, V., & Marco Valtorta. (2006). The Effects of Data Quality on Machine Learning Algorithms. ICIQ.

[5] Alding, O. (2018). Mobile Object Detection using TensorFlow Lite and Transfer Learning. Stockholm, Sweden: Digitala Vetenskapliga Arkivet.

[6] Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A Survey of Model Compression and Acceleration for Deep Neural Networks. IEEE Signal Processing Magazine Sessions, V., & Marco Valtorta. (2006). The Effects of Data Quality on Machine Learning Algorithms. ICIQ.

[7] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4510-4520.

[8] TensorFlow. (n.d.). TensorFlow Lite image classification Android example application. Retrieved from GitHub: https://github.com/tensorflow/examples/tree/master/lite/examples/image_classification/android