AI-Assisted Outcome Engineering for Mapping Natural Language to Radar Configuration Files

Darrell L. Young, Eric C. Larson, and Mitchell A. Thornton
Darwin Deason Institute of Cybersecurity
Southern Methodist University, Dallas, TX, USA
Emails: {dlyoung, eclarson, mitch}@smu.edu

***Abstract*—This paper shows how to use Large Language Models (LLMs) and DSPy prompt engineering to bridge the gap between expert knowledge and user natural language inputs. This approach improves radar system usability by leveraging the DSPy ReAct module and MIPROv2 prompt optimization to facilitate the dynamic generation of radar technical parameters.**

I. Introduction

Large Language Models (LLMs) can assist the engineer with the tedious task of creating structured data. For example, Table I shows the default radar configuration parameters for the Texas Instruments Radar Estimator tool for a short-range automotive radar board. This paper describes an experiment which uses an LLM to convert a natural language input like "Configure the radar to detect a pigeon." Once a radar configuration table has been obtained from the LLM it can be easily input into the Radar Estimation Tool [1]. The Radar Estimation Tool determines the feasibility of the desired parameters created by the LLM. This relieves a lot of technical burden from the LLM. The LLM's main purpose is to translate the user requirements expressed in natural language into technical requirement parameters expressed in the Radar Estimation Tool's engineering language. Use of the TI Radar Estimation Tool with the resulting LLM-produced radar configuration table will reveal if the technical parameters are achievable. For example, the LLM does not have to determine that the user's desire to detect sparrows at a range of a mile is not possible. The Radar Estimation Tool will show that. However, the LLM does have the burden of converting natural language to technical specifications for range, resolution, velocity, measurement rate, and so forth. In this paper we show an approach on how the LLM can use its function-calling skill to interpolate a Radar Cross Section (RCS) table based on bird weight to provide the important entry in the radar configuration table "Typical Detected Object (RCS m2). A similar LLM tool-calling approach can be used for other parameters or for more sophisticated approaches to estimating RCS.

The interpolation of bird RCS values, as shown in Table II, serves multiple purposes. First, it prevents the misclassification of our research as applicable to military applications. Second, it provides a straightforward method to validate radar configurations using commercially available Texas Instruments radar. Third, the interpolation approach is much less difficult than typical RCS electromagnetic modeling. Lastly, birds are an interesting use case in and of themselves for radar measurement.

TABLE I
Default Radar Configuration Parameters for IWR6843 Platform

Parameter	Value
Platform	IWR6843
Number of Rx	4
Number of Tx	3
Tx Gain (dB)	9
Rx Gain (dB)	9
Frequency Range (GHz)	60 - 64
Maximum Bandwidth (MHz)	4000
Tx Power (dBm)	12
Ambient Temperature (°C)	20
Maximum Detectable Range (m)	10
Range Resolution (cm)	5
Maximum Velocity (km/h)	26
Velocity Resolution (km/h)	2
Measurement Rate (Hz)	10
Typical Detected Object (RCS m^2)	2.0
Detection Loss (dB)	1
System Loss (dB)	1
Implementation Margin (dB)	2
Detection SNR (dB)	12

We discuss how our experiment applies to the concept of radar interpretability. In general, the conversion of natural language requirements to technical parameters has historically been accomplished using various rubrics where the user matches their specific use case to a table of similar use cases which are ordered and assigned a numerical value. Then, an equation is developed to translate technical parameters to the numerical rating. This approach suffers from the collapse of the technical multi-dimensional parameter space into the single dimensional rating scale. The LLM approach outlined in this paper does not have this limitation and thus has the potential to more accurately translate users' desires expressed in language to accurate technical parameters. In this study we focus on a user mainly interested in applying Texas Instruments radar products to detection and tracking of birds using the published RCS in Table II shown below.

The decision to focus on interpolating the RCS of birds stems from the need to develop radar systems capable of handling real-world environmental challenges in non-military applications. By choosing avian subjects, which frequently intersect with commercial radar detection zones, particularly

TABLE II
RCS AND BIOMASS OF VARIOUS BIRD SPECIES.
DATA ADAPTED FROM [2].

Bird	Biomass (kg)	RCS (cm^2)
European Blackbird	0.1	42
Chaffinch	0.024	2.2
House Sparrow	0.025	2.4
Lapwing	0.2	91
European Robin	0.017	1.1
Song Thrush	0.07	22
Starling	0.08	29
European Swift	0.043	8.3
Garden Warbler	0.02	1.7
Chiffchaff	0.008	0.3
Warbler	0.01	0.2
Sparrow	0.025	1.3
Plover	0.2	51
Pigeon	0.5	109
Duck	0.8	88
Swan	1	210
Eider	2.02	174
Brant	1.48	157
Lesser Snow Goose	2.63	190
White-fronted Goose	2.68	191
Long-tailed Duck	0.87	131
Pintail	0.95	135
Whistling Swan	6.8	261
Greater Snow Goose	3.07	200

in urban and suburban environments, we ensure our research remains applicable to civilian technologies. This choice also aligns with our goal to refine radar parameterization techniques that can be safely and publicly shared, avoiding any potential dual-use concerns that might arise with more sensitive applications.

Modern mmWave radar systems require careful selection of parameters such as transmit power, bandwidth, polarization, and antenna gains in order to detect targets with specific Radar Cross Sections (RCS). However, end-users often specify their needs in everyday language (e.g., "I need to detect a passenger vehicle at 70 m"), leaving system designers to manually compute or guess relevant radar parameters.

Additionally, the number of parameters to configure can be extremely large, from center frequency and bandwidth to idle times, slopes, sampling rates, power levels, and more. In these scenarios, *AI-powered large language models (LLMs)* can be invaluable: they can parse user requests, infer approximate values or compute precise values using tools, and recommend consistent sets of configuration parameters that comply with device and regulatory constraints. This significantly eases the burden on the radar engineer, who would otherwise be required to tediously navigate an overwhelming set of interdependent variables.

II. INTERPRETABILITY AND OUTCOME ENGINEERING IN RADAR SYSTEMS

Traditionally, interpretability in radar and imaging systems involves mapping high-level, real-world mission needs into technical parameters to ensure sensor or system efficacy. Tools like the National Image Rating Scale (NIIRS) and the General Image Quality Equation (GIQE) exemplify this approach by converting technical parameters into human-understandable ratings, thus linking performance to mission objectives [3], [4].

For example, in electro-optical imaging, a requirement to "take a picture of a bird at 50 yards for identification," translates through interpretability frameworks into specific camera settings like pixel density, focal length, exposure time, and aperture size, ensuring adequate resolution and contrast. Similarly, in radar systems, user specifications such as "detect a large truck at 100m" or "track a fast-moving drone at 200m" drive the system configurations to achieve necessary detection SNR, range and velocity resolution given the target RCS characteristics.

The Declarative Self-improving Python (DSPy), is a framework that transitions from traditional prompting to programming language models, enabling rapid iteration in building modular AI systems. It provides algorithms for optimizing prompts and weights across various applications, from simple classifiers to complex RAG pipelines and Agent loops. This approach promotes writing robust, compositional Python code, thus enhancing the model's ability to produce high-quality outputs [5].

Advancing to Outcome Engineering: While interpretability frameworks provide a foundation, the advent of Large Language Models (LLMs) and advanced tools like DSPy offer a more dynamic and expansive approach. This new paradigm leverages AI to interpret and translate complex mission needs into optimized system configurations across various domains more efficiently than traditional methods.

III. DSPY INTEGRATION AND ADVANCED PARAMETER OPTIMIZATION

DSPy, through its integration with LLMs, automates the generation of radar configuration parameters. The DSPy framework offers the `MIPROv2` optimizer, an advanced tool that enhances parameter tuning based on real-time data inputs and machine learning predictions.

An example screenshot in Section V of the TI mmWave Sensing Estimator illustrates the practical application of LLM-enabled advanced configurations, tailored through DSPy's AI-driven interface.

IV. RADAR THEORY AND TI ESTIMATOR

Texas Instruments provides reference documents and a Radar Estimator tool that leverage the fundamental radar range equation. The Estimator's web interface (Section V) allows the user to input parameters such as:

- **Number of Rx and Tx Antennas**,
- **Frequency Range** (e.g., 60–64 GHz),
- **Maximum Bandwidth**,
- **Transmit Power**,
- **Ambient Temperature**,
- **Typical Detected Object RCS**,

and outputs a device configuration (e.g., chirp start/end frequencies, sampling rates, number of loops).

A. Radar Range Equation Recap

We denote:

- P_t: Transmit power of the device,
- $G_{\mathrm{TX}}/G_{\mathrm{RX}}$: Transmit/Receive antenna gains,
- σ: Radar Cross Section of the target,
- λ: Wavelength,
- d: Range to target.

The received power P_{r} ultimately determines whether the radar can detect the target above the noise floor, factoring in system losses and required SNR margin [6], [7].

V. TI Sensing Estimator Screenshot

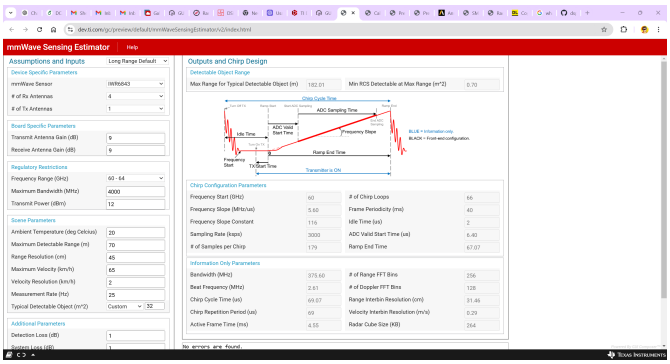

Fig. 1. Screenshot of the TI mmWave Sensing Estimator, showing how a user can set device parameters (left) and view the resulting chirp design (right). Full tool available at [1].

As shown in Figure 1, the Estimator interface [1] displays both a textual parameter list and a graphical chirp diagram. This interactive approach helps engineers understand the effects of each setting on range, velocity resolution, and other radar configuration settings.

VI. DSPy Integration for Automated Parameter Generation

A. Advantages of DSPy Prompt Optimization

DSPy's prompt optimization provides a principled approach to prompt engineering for Large Language Models (LLMs). Distinct from methods requiring extensive model fine-tuning and large datasets, DSPy efficiently leverages the intrinsic capabilities of LLMs to comprehend and produce intricate configurations using only minimal exemplars.

Efficient and Dynamic Parameter Optimization: DSPy's optimization engine is specifically designed for prompt engineering, using a few carefully selected examples to guide LLMs toward generating optimal outputs. This method substantially enhances operational efficiency by:

- **Minimizing Resource Consumption:** DSPy reduces the need for large training sets and computational resources, facilitating quick deployment and agile modifications crucial in fast-paced engineering environments.
- **Maximizing Adaptability:** Allows engineers to swiftly adjust system specifications or respond to new requirements without necessitating a complete model retraining, thus providing exceptional flexibility.

Capability to Invoke Tools and Extract Parameters: The DSPy ReAct module further extends the utility of prompt optimization by:

- **Integrating External Tools:** ReAct can seamlessly invoke external computational tools or software functions within the LLM framework, enabling complex data processing and analysis tasks to be executed directly from the LLM prompts.
- **Extracting and Inferring Parameters:** LLMs within DSPy are adept at analyzing textual descriptions or structured data to extract and infer necessary technical parameters, translating abstract requirements into concrete system configurations.

Reduced Need for Model Fine-Tuning: DSPy's approach circumvents the conventional necessity for extensive fine-tuning:

- **Lowering Technical Barriers:** Organizations can leverage state-of-the-art LLM capabilities without deep expertise in AI, democratizing access to advanced technologies.
- **Ensuring Model Stability:** The prompt-based method maintains effectiveness over time without frequent retraining, contrasting with finely tuned models that often require continuous updates.

Practical Impact and Applications: DSPy's practical applications demonstrate significant enhancements in operational settings:

- **Accurate Optimization:** DSPy fine-tunes radar system parameters like detection range and resolution more precisely than traditional methods.
- **User-Centric Design:** Simplifies interactions for non-expert users, allowing them to define desired outcomes instead of intricate technical details.

DSPy's prompt optimization capabilities provides a new approach in converting natural language requirements to technical parameter specification. Future explorations will aim to broaden these methodologies to wider applications, affirming DSPy's pioneering role in AI-driven system configuration.

VII. Experimental Methodology

This study uses an experimental setup shown below in Figure 2.

A. Workflow Description

The experimental setup is comprised of the following key components and data flow:

1) **OpenAI GPT-3.5 Turbo:** The process begins on a notebook computer where we utilize OpenAI's GPT-3.5 Turbo to generate training examples. This model provides high-quality, contextually relevant data that forms the basis for our experiments. OpenAI was used for convenience. GPT-3.5 Turbo Model is inexpensive yet able to generate simulated natural language inputs and matching formatted radar configuration structures

using the initial Texas Instrument's default configuration files as examples.

2) **Data Transfer to University Superpod:** The generated training examples are then transferred to the University SuperPOD. This high-performance computing environment is equipped with the Ollama framework, which houses the Microsoft Phi-4 model.
3) **DSPy Optimization:** Within the University Superpod, the DSPy optimization tool refines and optimizes the radar configuration prompts using the Chain-of-Thought (CoT) module for the variable parameters and the ReAct modules with the RCS interpolation tool for the Typical Detected Object parameter.
4) **Testing Phase:** The optimized prompt program is used to give the LLM necessary context data to process the test prompts.
5) **Results Collection:** The outcomes of the tests are collected and sent back to the notebook to be included in this publication preparation.

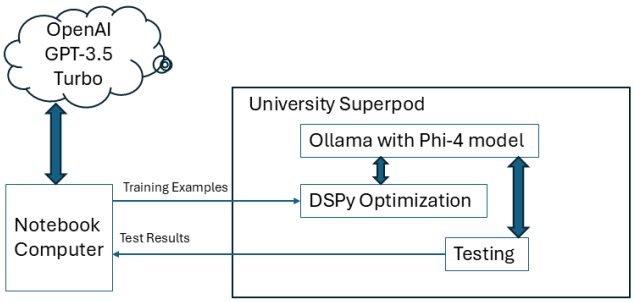

Fig. 2. Diagram of the experimental setup showing the workflow from training example generation to testing the optimized prompts.

The workflow resulted in a radar configuration output compatible with the Texas Instruments Radar Estimation Tool input configuration format. Below in Table III is the sample output for the Falcon input question, "Configure the radar to detect a falcon, which has a biomass of 2.5 kg. The system uses the IWR6843 platform."

TABLE III
RADAR CONFIGURATION PARAMETERS FOR DETECTING A FALCON

Parameter	Value
platform	IWR6843
num_rx	4
num_tx	3
tx_gain	9
rx_gain	9
frequency_range	60 - 64
maximum_bandwidth	4000
tx_power	12
measurement_rate	10
detection_loss	1
system_loss	1
implementation_margin	2
detection_SNR	12
ambient_temperature_degC	20
maximum_detectable_range	50
range_resolution	2
number_of_objects	15
nearest_object_spacing	0.5
maximum_velocity_kmph	40
velocity_resolution_kmph	1
typical_detected_object	0.018659016393442623

The focus of this study is the LLM tool invocation of the RCS interpolation function. The Table IV shows the estimated output RCS for the various test cases, one of which includes the mythical Kraken, which was mentioned as part of the user's conversational input. Because the Kraken weight was given in pounds, the LLM ReAct agent converted the weight to kg and called the interpolation function. The entries with question marks were not provided a weight in the input query. For these cases, the LLM had to use its internal knowledge to estimate the biomass.

TABLE IV
RCS ESTIMATION TEST RESULTS FOR RADAR CONFIGURATION

Idx	Detection Query	RCS (m^2)	Comments
1	Robin, 0.065 kg	0.001946	Precision for small birds.
2	Kraken, 1500 lbs	2.4681	High biomass, mythical creature.
3	Nightingale, ? weight	0.00022	Biomass estimated by system.
4	Eagle, 6.5 kg	0.02561	Accurate for well-known birds.
5	Sparrow, ? weight	0.00022	Estimates biomass for common small birds.
6	Hummingbird, ? weight	0.000055	Effective for very small birds.
7	Flamingo, 4 kg	0.02152	Handles moderately sized birds accurately.
8	Peacock, 12 lbs	0.03460	Converts lbs, calculates RCS for medium bird.
9	Swan, ? weight	0.03460	Biomass estimation for larger birds.
10	Falcon, 2.5 kg	0.01866	Precise RCS for birds of prey with known weights.

B. Implementation Details

The experimental setup leveraged two critical components of the DSPy framework: the `CoT` (Chain of Thought) module and the `ReAct` module equipped with the Bird RCS interpolation tool. Each module was tasked with specific aspects of the radar configuration to optimize the system's performance without overloading the computational resources.

1) Chain of Thought (CoT) Module: The DSPy `CoT` module was responsible for dynamically setting several critical radar parameters based on the input requirements and environmental conditions. These parameters included:

- Ambient temperature in degrees Celsius (`ambient_temperature_degC`): 20,
- Maximum detectable range in meters (`maximum_detectable_range`): 10,
- Range resolution in meters (`range_resolution`): 5,
- Maximum velocity in km/h (`maximum_velocity_kmph`): 26,
- Velocity resolution in km/h (`velocity_resolution_kmph`): 2.

These settings were crucial for ensuring the radar system's adaptability to various operational conditions and were derived based on the scenario descriptions provided by the user.

2) ReAct Module with Bird RCS Interpolation Tool: In contrast, the `ReAct` module focused on estimating the Radar Cross Section (RCS) for the birds under observation. This estimation was crucial for accurately configuring the radar to detect different bird species effectively. The RCS values, dynamically computed by the interpolation tool based on the biomass data, were integrated into the radar's configuration parameters.

3) Fixed Parameter Configuration: To simplify the initial experimental setup and avoid the complexities of managing an extensive set of variable parameters, several radar settings were kept fixed. These included:

- Radar platform: IWR6843,
- Number of receivers (`num_rx`): 4,
- Number of transmitters (`num_tx`): 3,
- Transmitter gain (`tx_gain`): 9,
- Receiver gain (`rx_gain`): 9,
- Frequency range (GHz) (`frequency_range`): "60 - 64",
- Maximum bandwidth (MHz) (`maximum_bandwidth`): 4000,
- Transmit power (`tx_power`): 12,
- Measurement rate (`measurement_rate`): 10,
- Detection loss (`detection_loss`): 1,
- System loss (`system_loss`): 1,
- Implementation margin (`implementation_margin`): 2,
- Detection Signal-to-Noise Ratio (`detection_SNR`): 12.

These parameters were selected for their stability and reliability in typical radar operation scenarios, ensuring a controlled environment for testing the effectiveness of the DSPy optimizations.

C. Implementation Resources: phi-4 Model and University SuperPOD

Our DSPy-based pipeline runs on top of large language model (LLM) capabilities to parse user queries and generate radar configuration files. In particular, we used **phi-4** [8], a 14-billion parameter language model which incorporates synthetic data throughout training. Despite its partial distillation from GPT-4, phi-4 excels on reasoning-centric tasks due to improved data generation and post-training techniques.

All optimization experiments (e.g., parameter tuning with MIPROv2) were conducted on the **Southern Methodist University NVIDIA DGX SuperPOD** [9] resource, a high-performance computing cluster with 20 DGX A100 nodes. Each node has 8 GPUs, enabling large-scale parallel evaluations of different prompt or instruction sets. This setup allowed us to rapidly converge on an optimal prompt strategy for interpretability queries, with minimal iteration time for interpretability queries.

Table IV illustrates how natural language queries from a user are mapped to structured output, `typical_detected_object`. Our prompt optimization method leverages DSPy's `MIPROv2` module to explore various instruction and few-shot example combinations, converging on a prompt strategy that produces consistent, structured responses. The same approach can be expanded to include additional radar parameters, such as `range_resolution`, `velocity_resolution`, or `detection_SNR`. Rather than outputting a single `typical_detected_object` field, the prompt optimization could guide the LLM to return multiple structured fields (e.g., JSON objects containing both `target_RCS`, `max_range`, `probability_of_detection`, etc.). This scalable method allows domain experts to iteratively refine prompts and few-shot examples to produce rich, multi-parameter data suitable for more advanced radar planning and configuration pipelines.

VIII. Conclusion

This paper has introduced Outcome Engineering as a transformative approach to system configuration, leveraging the capabilities of Large Language Models (LLMs) to enhance radar system applications and other technology-driven operations. The LLM-based Outcome Engineering approach helps customize the radar configuration.

IX. Future Work

We plan to continue to develop and test AI-ML technology using the Cyber Autonomy Range (CAR) which is a dedicated facility designed to assess the resiliency, reliability, and cybersecurity of autonomous systems [10]. The CAR provides a secure environment to simulate or emulate external conditions while subjecting the AS to various forms of cyber-attacks. By instrumenting the decision-making processes of AS (especially

ML/AI subsystems), the CAR can identify vulnerabilities in sensor data processing pipelines.

Recent work has expanded the CAR's scope by incorporating *RF situational awareness* into robotic simulation systems [11]. This addition allows the detection and localization of radio emitters in the environment, enabling algorithms for collision avoidance that rely on RF sensors. The approach emphasizes privacy, security, timeliness, and safety, supplementing existing sensor suites like RGB cameras, LIDAR, radar, and IMU devices.

When radar detection is central to an AS, accurate RCS values are crucial for performance modeling. The CAR's realtime 3D motion simulations use the aspect angles of the target to lookup the RCS values pre-computed by Ansys HFSS. HFSS SBR+ module computes monostatic/bistatic RCS and enables radar signature imaging (e.g., range profiles, ISAR). In the CAR context, these RCS models can be dynamically updated if the target's shape, orientation, or reflectivity changes—or if a cyber-attack spoofs sensor data.

The fusion of LLM-driven parameter configuration with physics-accurate HFSS simulations ensures that mission-level requirements are precisely translated into practical, workable radar configurations. This synergy between theoretical concepts and practical applications paves the way for future innovations in system design and underscores the potential of Outcome Engineering to revolutionize how we understand and implement technology in response to complex challenges. An interesting possibility is the use of machine learning to interpret and refine radar measurements to update object properties. For example, in the case of avian targets, collected radar data could be used to better estimate biomass, Radar Cross Section (RCS), and other target properties, which could then be queried by the LLM using natural language to configure the radar for additional collections.

Future work will focus on expanding the scope and depth of this research by:

- **Advanced NLP Methods**: Refining the interpretability and effectiveness of Outcome Engineering by handling ambiguous or conflicting user inputs and further enhancing the metrics for diverse missions.
- **Deeper CAR Integration**: Conducting large-scale tests with multiple radar-equipped autonomous systems under coordinated cyber-attacks to validate and improve the robustness of the proposed models.
- **Robust Experimental Benchmarks**: Moving beyond simulations to conduct real lab and field experiments that capture complex scenarios involving multipath effects, clutter, and adversarial interference, aiming to establish more robust and resilient system configurations.

A more comprehensive discussion of these topics will be detailed in our forthcoming journal publication, which will explore the extensive capabilities and broader implications of AI-assisted Outcome Engineering in the context of next-generation technological solutions.

REFERENCES

[1] "TI mmWave Sensing Estimator," https://dev.ti.com/gc/preview/default/mmWaveSensingEstimator/v2/index.html, accessed: 2025-01-23.

[2] S. A. Gauthreaux Jr, A. M. Shapiro, D. Mayer, B. L. Clark, and E. E. Herricks, "Detecting bird movements with l-band avian radar and s-band dual-polarization doppler weather radar," *Remote Sensing in Ecology and Conservation*, vol. 5, no. 3, pp. 237–246, 2019.

[3] D. L. Young and T. Bakir, "Cognitive modeling to predict video interpretability," in *Proc. SPIE 8053, Geospatial InfoFusion Systems and Solutions for Defense and Security Applications*, 2011, p. 80530M.

[4] W. Schwartzkopf, J. Brown, G. Farquharson, C. Stringham, M. Duersch, and J. Heemskerk, "Radar generalized image quality equation applied to capella open dataset," in *2022 IEEE Radar Conference (RadarConf22)*. New York, NY, USA: IEEE, 2022, pp. 1–5.

[5] O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, S. Haq, A. Sharma, T. T. Joshi, H. Moazam, H. Miller, M. Zaharia, and C. Potts, "Dspy: Compiling declarative language model calls into self-improving pipelines," 2024.

[6] M. I. Skolnik, *Introduction to Radar Systems*, 3rd ed. McGraw-Hill, 2001.

[7] M. A. Richards, *Fundamentals of Radar Signal Processing*, 2nd ed. McGraw-Hill, 2014.

[8] M. Research, "Phi-4: A 14-Billion Parameter Language Model," https://www.microsoft.com/en-us/research/uploads/prod/2024/12/P4TechReport.pdf, 2024, accessed: 2025-01-23.

[9] Southern Methodist University, OIT Services.

[10] D. L. Young, M. Bigham, M. Bradbury, E. Larson, and M. Thornton, "SMU-DDI Cyber Autonomy Range (CAR): Incorporation of Resiliency, Reliability, and Cyber Security in Autonomous Systems," in *2022 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*. IEEE, Oct 2022, pp. 1–5.

[11] S. Gibbs, M. A. Thornton, and D. L. Young, "Adding RF situational awareness to robotic simulation systems," in *Proc. SPIE 12540, Autonomous Systems: Sensors, Processing, and Security for Ground, Air, Sea, and Space Vehicles and Infrastructure*, June 2023, p. 1254009, published: 13 June 2023.