Integrating Software and Systems Engineering: A Framework for Human-Centric Design of Augmented Reality Applications

Emi Aoki*, Bach Tran†, Flore Norcéide*, Vinh Tran*, Shivakumar Sastry‡, Charles Thompson*, and Kavitha Chandra*
*University of Massachusetts Lowell, Lowell, USA
†Dassault Systemes Americas Corp., USA
‡Dayananda Sagar University, Bengaluru, India

***Abstract*—Modeling from the perspectives of software engineering and systems engineering have co-evolved over the last two decades as orthogonal approaches. Given the central role of software in modern cyber-physical systems and the increasing adoption of digital engineering practices in complex systems design, there is now significant opportunity for collaborative design among system users, software developers, and systems engineers. Model-based systems engineering (MBSE) and systems modeling languages can support seamless cross-domain connectivity for design, simulation, and analysis of emerging technologies such as Augmented Reality (AR). This paper presents a co-design process for extending the capability of an existing AR application referred to as a No-Code AR Systems (NCARS) framework. NCARS enables content developed by multi-domain authors to be deployed on AR devices through a software layer that bridges the content to the game engine that drives the AR system. Utilizing a software dependency diagram of the AR Annotation function, an existing MBSE model of the AR system is extended to include the structure and behavior of relevant software components. This allows a modular design of the system to address needs in integrating new requirements into the existing application. New user requirements for tracking items in motion in the user's physical environment with virtual annotations in the augmented space are collaboratively designed and visualized through use case, block definition, internal block, and sequence diagrams. They capture the required structure and behavior of the proposed to-be system.**

***Index Terms*—Augmented Reality, Digital Transformation, Dynamic Annotation, Model-Based Systems Engineering, Item Tracking, Software Engineering**

I. Introduction

As immersive experiences with augmented reality (AR) and virtual reality (VR) devices become more accessible, there is a growing need for the design of applications that will support more adaptable and customized human-technology interactions with such systems. This is echoed in the premise of emerging Industry 5.0 that expects industrial work contexts and environments that have adopted the automation and data-driven technology of the last decade to ensure the well-being of workers who will interface with this technology [1]–[3]. This goal includes leveraging the resilience of humans in utilizing technology to enhance their skills, collaborative capacity, and build safer work environments and practices. Grech et al. [4] examine the potential of artificial intelligence (AI) and VR technologies for product ideation in collaborative spaces. The authors demonstrate that an AI-supported VR environment facilitates brainstorming, creativity, and immersion of team members in the design process. It enhances focus and interpersonal relationships, ultimately improving product creation and collaboration. Quandt et al. [5] present a user-centered design approach for evaluating an AR-based assistance system. They develop an AR system to assist users in maintaining heating, air conditioning, and cooling systems. The authors demonstrate that the integration of user-based evaluations throughout development reduced users' frustration in interacting with the new system from 35 to 27.22 on the grading scale. Villani et al. [6] evaluate an adaptive industrial automation system, INCLUSIVE, through three real industrial use cases with 53 shop workers. The framework assessment considers performance metrics that include worker satisfaction and usability. Their results indicate that 79% of workers are satisfied with the human-machine interface and the task completion time is reduced by 36.9%. Although some challenges are associated with user-technology interactions, the methodology presented by these studies show that AR systems can be iteratively refined based on practical user experiences to align with the principles of Industry 5.0: emphasizing the seamless integration of human and technological elements.

The integrated system comprising the AR device, the physical environment of the user, associated data networks, sensors, embedded systems, and computing infrastructure can be recognized as a class of cyber-physical systems (CPS) that also include humans in the loop. The use of AR systems has been increasing across industrial, health, and manufacturing domains [7]–[9]. For example, there is a tremendous opportunity for immersive technologies to provide point of care diagnostics across a spectrum of scenarios, such as during surgical operations [10], in clinical practices [11], nursing [12], and health education [13]. The humans in the loop include, in addition to AR system users, other decision-makers who are often interested in better understanding if these digital technologies are fit for purpose and their flexibility for improving the efficiency of relevant operations in their organization. The well-being of health professionals and their clients is of paramount importance, and to support this, it is critical that as digital transformation takes place across the workplace, a systems and design thinking process is integrated for addressing specific needs of individuals and units that are adopting these technologies [14].

Model-Based Systems Engineering (MBSE) has the potential for the various partners involved to engage in the design of AR applications and contribute to new capabilities required by the system users. MBSE and Systems Modeling Language (SysML), which is a graphical language for MBSE designed to extend the Unified Modeling Language (UML) for software specification, has been applied to describe the requirements, structure, and behavior of complex systems. The potential of SysML for describing the software that drives these systems in a modular approach is explored in this work. As noted by Hause and Thom [15], by modeling software specification using SysML, software engineers can benefit from the system model context, offering a level of rigor above what can be achieved in standard software engineering approaches. This paper demonstrates this advantage using the example of applying the AR Annotation function for a new requirement and discusses a framework for systematic integration of software engineering models in the MBSE framework.

The paper is organized as follows. A brief background of the AR model-based system design using SysML is presented in Section II. Section III discusses a software conceptual dependency diagram for high-level mapping of the software artifacts to a SysML-based representation. Section IV extends our prior SysML model of the AR system to include essential software dependencies and enable efficient design of new capabilities. An example of how such an extended software and system model can enable agile software design is presented in Section V. Section VI summarizes the paper and discusses future work in democratizing human-centric AR design.

II. Background

In our prior work [16], [17], a systems-level description of the AR application design and deployment for a table-top conveyor system shown in Fig. 1 was presented using SysML. The AR application design was referred to as a No-Code AR Systems (NCARS) framework, in keeping with the objective of enabling users across multiple domains to create their applications of interest and deploy them on the AR device without the need for extensive coding expertise. In this work, we extend this model to include the design of the underlying software artifacts that enable NCARS to deploy user-generated applications. When a user requires additional capability of the AR system to support their application, software developers can benefit from both a system and a software model to efficiently extend the code to support the required capability.

The conveyor system is equipped with ferrous and non-ferrous metal detector sensors placed at specific locations to detect the presence of metal pegs and plastic washers. The system consists of continuous belt loops that move across the system, terminating in three different outlets. The integration of AR devices into the conveyor system enhances the operation efficiency by providing the operator with real-time information and data about its various components. The AR augmentation provides seamless monitoring and management of the material in transit, allowing quick decisions and interventions.

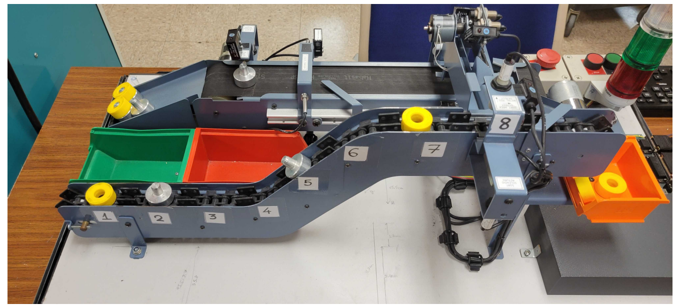

Fig. 1. Conveyor belt system

A high-level use case of NCARS with its as-is capability is shown in Fig. 2. NCARS supports the four activities of Annotation, Guidance, Navigation, and Safety. In *Annotation*, the user can request just-in-time information of physical items of interest. With *Guidance*, the user is presented with virtual information bubbles that contain information to complete a sequence of tasks. The user may choose *Navigation*, which presents virtual navigational signs such as arrows that support them in navigating a physical environment. Through *Safety*, the user will be informed of the status of various safety parameters and alarms.

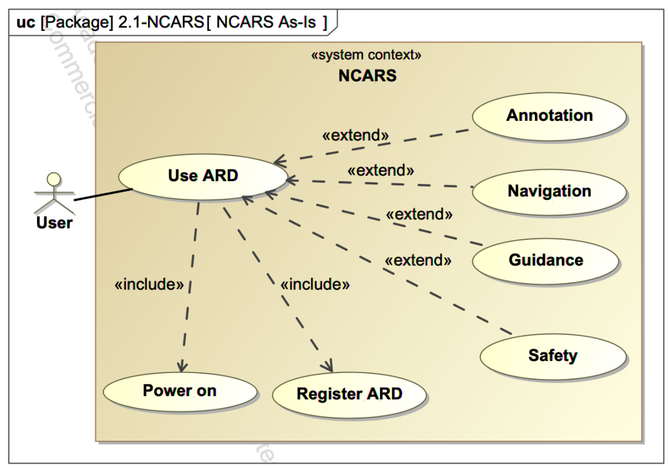

Fig. 2. Primary (As-Is) capabilities of NCARS

NCARS has been designed within the larger context of an application stack. A game engine such as Unity [18] or Unreal [19] is the bottom-most layer and provides the rendering capability for the application. It renders multi-model graphics and audio. NCARS rides on the game engine and simplifies the use of the game engine for the supported four activities. It offers a more restrictive interaction than a general game but makes it easier for application development. On top of NCARS is the Human-Centrc design layer, where content from domain experts who are using the AR device is generated and managed by the NCARS layer. This content can be in the form of documents, audio, video, or other domain-specific artifacts that are required to be produced in the augmented space.

NCARS uses a registration method presented in [20]. The registration method and NCARS architecture restrict the annotation to prespecified fixed locations on the conveyor system in the AR field of view (FOV). This paper considers an extension of NCARS that will allow the annotation of items in motion, i.e., pegs and washers, that move on the conveyor system in the FOV.

By extending the MBSE model of the AR system to include software dependencies for functions such as Annotation, programmers working at the NCARS layer are provided the structure and behavioral models of the software that can be extended to support new capabilities of the AR system.

III. AR Application Software Structure

This section discusses the software design of the Annotation function in NCARS. This function allows the system to present annotations in the FOV that have useful information relating to the user's point of interest. The term *object* will refer to an instance in the software engineering context, and the term *item* will refer to any part or component that the user interacts with in the physical world. *Artifacts* may refer to annotations in the augmented space, such as an information bubble, an audio clip, or an image presented to the user. The user interacts with this artifact by accepting the cue and may, in response, affect actions in the physical world via some physical item. For example, an annotation may instruct the user how to reset the system, and in response, the user may press and hold the reset button.

The software architecture for the Annotation system in NCARS is conceptualized and presented as a high-level class diagram in Fig. 3. This diagram emphasizes the aspects of the software that acquires and presents the annotations in the FOV and attaches them to the corresponding physical items in the user's environment. This diagram is a typical class diagram in object-oriented design where each block defines a class or an interface. Classes are the template definition of a design element and represent the elements that are changeable by the software designer, whereas the interface blocks represent elements from external systems whose behaviors are exposed through Application Programming Interfaces (APIs) and are normally considered as black-boxes. The properties and behaviors of the design element are represented by the block's properties and methods, respectively. They have a form of <property: type> and <method(): return type>. The return type of a method may be omitted if no element is returned. The solid line arrows between class blocks represent an association relationship. This indicates that an instance of the class at the arrow's tail references the instance of the class to which the arrow is pointing, allowing access to their methods and attributes. The value at the arrowhead (e.g., 0..1) specifies the limit of permitted connections. The dashed line arrows from one class block to an interface block represent the dependency relationship, indicating that the internal structure of one class relies on the given interface.

The central class is *AnnotationController*, which updates and controls the life cycle of all virtual annotations. The controller depends on the AR device's interface for the inputs, which are the user's hand gestures, voice commands, and the embedded camera's direction. An interface to the external Computer-Aided Design (CAD) server is required for ray casting [20] to detect if the user is looking or pointing at an item and to acquire the position of a known item. The controller obtains the server information from *AppConfig*, including IP addresses and API interfaces, and the communication is carried out asynchronously. Virtual annotations will be generated and managed by the controller when the user's hand or AR device camera intersects with pre-defined items in the physical environment. They are rendered in the form of information bubbles and instruction panels. The former is generated directly inside the controller, while a reference to the *InstructionPanel* class is used to reduce the workload of the controller class. The virtual annotations carry information about the item and are placed at the exact position of the item. The other classes in the diagrams are used to represent either a data type or information exchanged between the classes.

Although this class diagram is a convenient visual tool in object-oriented design, it lacks several behavioral details that can be better represented in SysML. To demonstrate these benefits, we translate and improve the design in a systems modeling framework. The migration to SysML also brings traceability and error detection capabilities that help to scale up the design and extend the framework with new applications.

IV. Extending the AR System Model with Software Dependencies

This section illustrates how SysML Block Definition (BDD), Internal Block (IBD), and Sequence diagrams serve to show the structure and behavior of various interacting components in the NCARS system context, in more detail than a typical object oriented design model.

The high-level system structure is depicted in Fig. 4. This includes a layer of the AR application hardware and software, the physical system, the communications network that interconnects all components, and the AR application data that includes artifacts required for the application. The AR hardware includes devices, such as the Microsoft® HoloLens and Magic Leap®, and various servers for monitoring, computing, and databases that store CAD models of the physical system and application-specific data. The AR system users are also included in this structural diagram, capturing the various users that potentially interact with the system. Blocks within the red dashed rectangle and thicker borders show new components added to the as-is system as a proposed solution to extend its capability. The use of these new components is further described in Section V.

The relationship between the blocks is captured by the specific direction and shape of the arrows that connect the blocks. A solid line with a filled diamond arrow represents

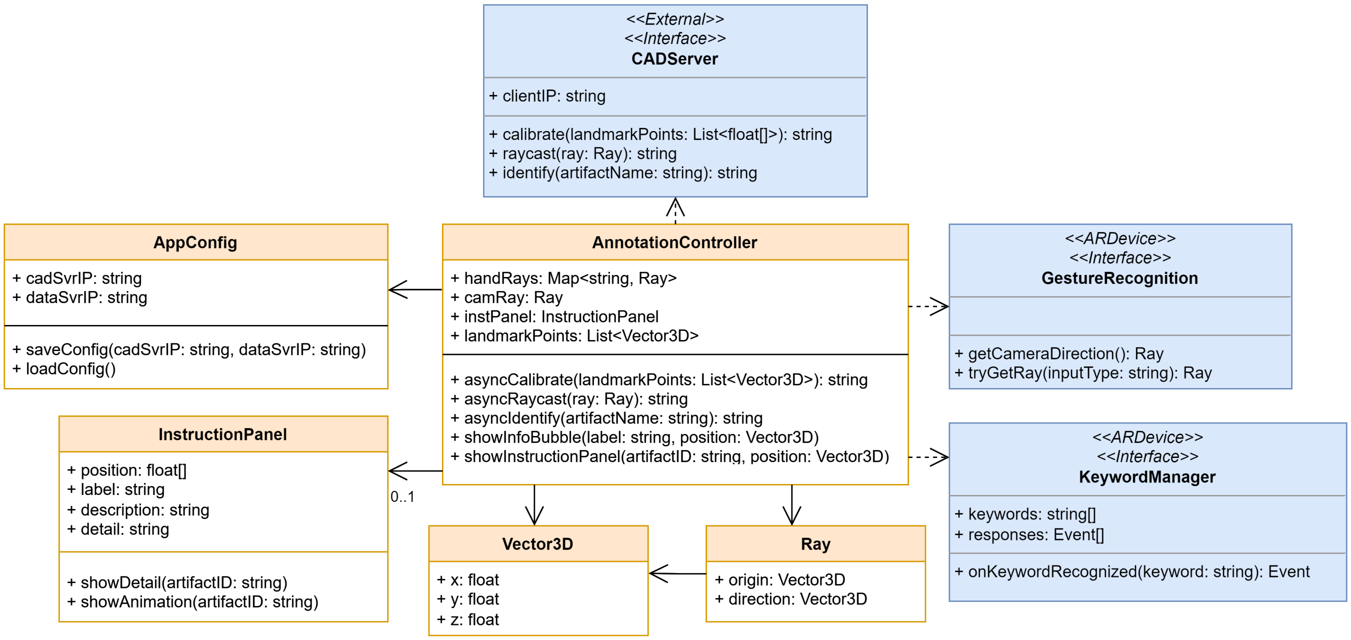

Fig. 3. The conceptualized class diagram of the Annotation Function in NCARS

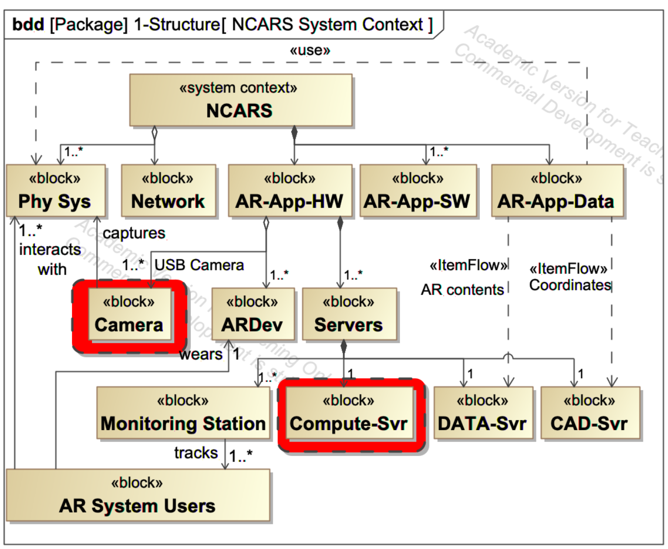

Fig. 4. High level NCARS system structure

a direct composition, indicating that NCARS is composed of AR-App-HW, AR-App-SW, and AR-App-Data. The Servers block is composed of the Monitoring Station, Data, CAD, and Compute servers. If child blocks can exist independently of the parent, a directed aggregation is shown as a solid line with an open diamond. This is the case with physical systems and networks, and the relation between AR-App-HW, AR devices, and USB Cameras. A solid line with an open arrowhead represents a unidirectional association with another block, as seen in the AR system users' interactions with the NCARS system for particular purposes, noted against the arrow. The type of dependency between blocks is indicated by the dotted line with the open arrow. *use* indicates that a block utilizes another whereas *Item Flow* specifies items being sent from one element to the other. Fig. 5 elaborates on the AR contents and coordinates *Item Flow*, describing the data types store in Data and CAD servers. The data may correspond to the contents generated by domain experts as described in Section III. The solid line with a closed arrowhead in this figure shows the generalization relationship, e.g., Animation is the sub-type of 3D AR content.

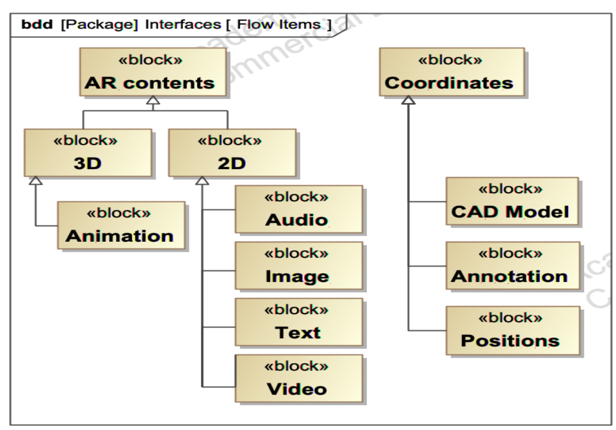

Fig. 5. Components saved in DATA and CAD servers from Data sources

The AR-App-SW block is expanded in Fig. 6. While Fig. 3 focuses on the Annotation features, Fig. 6 includes software assets that support NCARS for the four tasks. Blocks represent classes or sub-classes in the software, and each block is further classified into domain or functional classes. Domain class blocks define attributes and methods specialized to each of the activities that NCARS supports, whereas functional class blocks define supporting features that are reusable by multiple domain classes and also act as an interface to convey user inputs to the software application.

The orange-colored, thinner highlighted boundaries indicate that the object instances corresponding to these classes would be executed once in the application. In contrast, object instances shown without these boundary highlights have functionalities that are executed multiple times until the object instance is inactivated.

Note that the *GestureRecognition* and *KeywordManager* interface classes in Fig. 3 belong to the *User Input Manager* block. The *AppConfig* and *InstructionPanel* classes that interact with the *AnnotationController* class are also included, along with other classes such as *Registration* and *CalibrateMenu*, which are required for NCARS.

A block inside the red dashed rectangle denoted with thicker boundary emphasizes a new class to be added to extend the AR system's as-is capability. Section V further describes this new capability design.

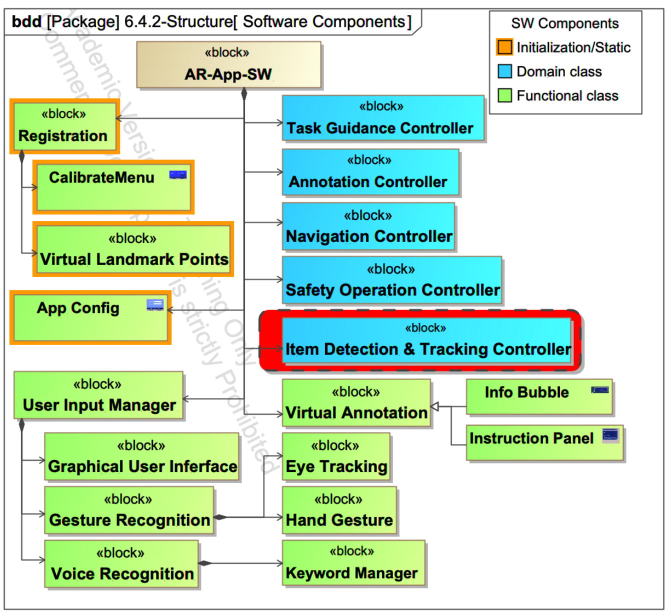

Fig. 6. High-level AR application software structure

With the high-level structure constructed, each of NCARS functions will be further described. Fig. 7 shows the Annotation as-is system. This is equivalent to the translation of the conceptualized software diagram shown in Fig. 3, excluding the interfaces. The Annotation Controller is composed of a Ray with data type of Vector 3D, which is a reusable component. The controller associates with User Input Manager and App Config to perform the required operations. The two properties with data type IP Address in the App Config block specify the servers' addresses in the network environment. Also, the controller manages the Virtual Annotation.

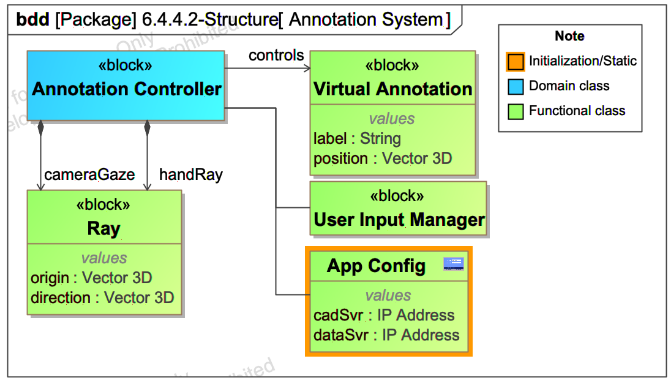

Fig. 7. Annotation as-is system structure

The particular detail, i.e., specifying what is being exchanged via object interactions is abstracted from software models and naturally represented in IBD diagrams. This level of detail, and other capabilities of the MBSE framework, can be leveraged to engage systems and software engineers in extending system capabilities. The IBD in Fig. 8 depicts how the components in the NCARS system interact to perform the Annotation feature. The small squares attached to the border of blocks are ports, which are points where the component and external entities interface. A line connector between two ports conveys the transfer of data or objects. The Annotation Controller block inputs user actions such as voice or gestures. Through the controller, the CAD server receives these inputs to process the user requests for specific annotation labels. The controller then initiates or updates the virtual annotation with this information from the CAD server.

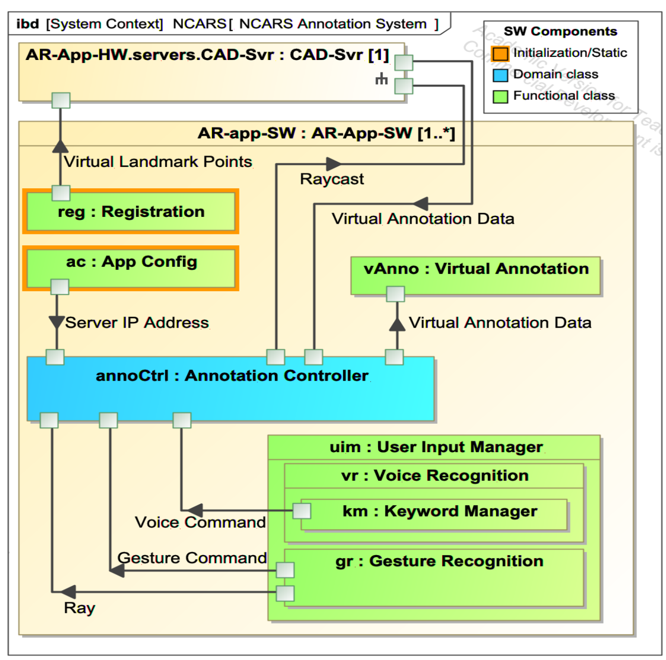

Fig. 8. Annotation as-is system connection

The sequence diagram shown in Fig. 9 provides further detail on the dynamic behavior of the components described in Fig. 8, depicting the operations that are invoked sequentially. An open arrow indicates an asynchronous operation whereas a closed arrow shows synchronous behavior. A response is denoted by a dashed line. The first three messages represent the sequence of behaviors invoked during the registration process. The Calibration operation is only initiated when the user chooses to generate new virtual landmark points. This operation is required when the user starts the application and the AR device is at a new position relative to the physical system, i.e., the table-top conveyor belt. To account for the time duration until the server completes the computation process, the Calibration operation is called asynchronously. When the Annotation feature of NCARS is activated without any error in the registration process, the rest of the interactions are invoked when the guard condition (viz. timeDiff and rayDiff) is met. The user inputs, i.e., ray, are sampled at intervals of one second when an angle difference between successive rays are smaller than five degrees. If the ray intersects with an item, virtual annotations will be generated. These interactions are repeated until the user terminates the application.

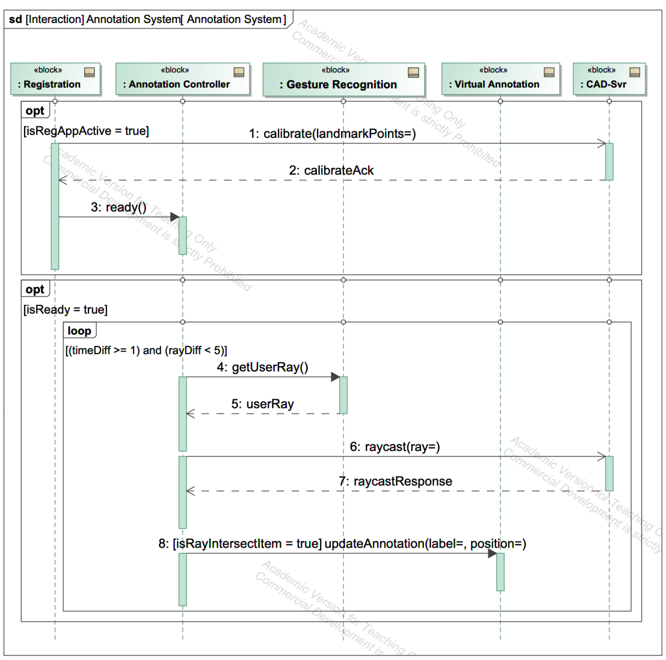

Fig. 9. Annotation as-is system interaction shown in sequence diagram

Representing the structure and behavior of the software architecture in SysML helps inform where a new capability can be added and which of the components would be affected or reused. The next section describes the design of additional components to achieve an extension of the system.

V. Adding New AR Capability: Object Detection and Tracking in Augmented Space

This section discusses the approach to efficiently extend the as-is system model described in the previous section with a new AR application capability required by a user. The new requirement is that a moving object in the physical space is to be augmented with a new annotation label each time it changes position. It includes tracking items of interest, classifying them, and annotating their positions as they move with new virtual information labels. Such applications are important in several domains such as an assembly line worker following annotated instructions as an object they are working on moves to a new station or a physical therapist requiring their client to follow a moving object with hand, head, or eye movements.

The new AR application capability is presented using an updated use case diagram shown in Fig. 10. This extends the use case diagram shown in Fig. 2 with a new function denoted Tracking, which is shown to include functions for the user to select items in the physical space that are to be tracked.

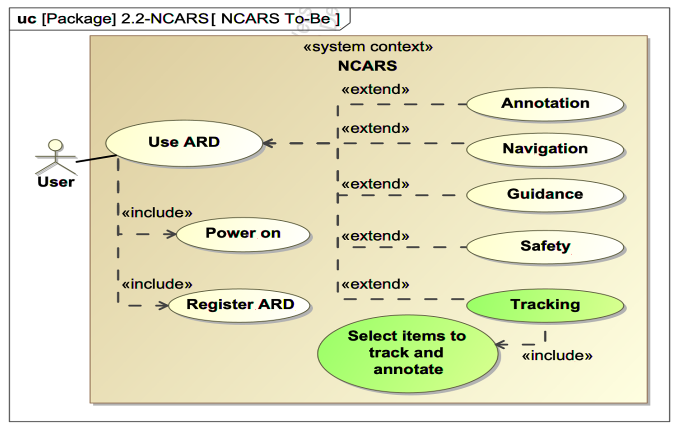

Fig. 10. Use Case describing a new capability of NCARS

The new functionality can be achieved by various methods such as sensor- or vision-based approach. The architecture demonstrated in this paper utilizes streaming inputs from an external camera for detection and tracking. The *to-be* system is designed based on this approach. The software developer will need to integrate a new class in the software that is denoted as Item Detection & Tracking Controller in the IBD shown in Fig. 11. This class extends the Annotation functionality with new operations of creating a position-dependent label and augmenting the physical space with this label.

The *to-be* system shown in Fig. 11 is designed to have a similar structure as the Annotation *as-is* system described in Fig. 8. This is to leverage the use of existing functionality and minimize the needs of additional components to be added to achieve agile development. New components are highlighted by the color blocked (red) rectangles. The Item Detection & Tracking Controller receives data on the user's items of interest and sends it to the CAD server. The compute server receives the target items from the CAD server and outputs the results of the detection and tracking operations. The CAD server saves and translates the data to virtual coordinates. The controller will then receive the information and annotate the virtual space. Once the target item is selected, the compute server continuously outputs positional data to the CAD server

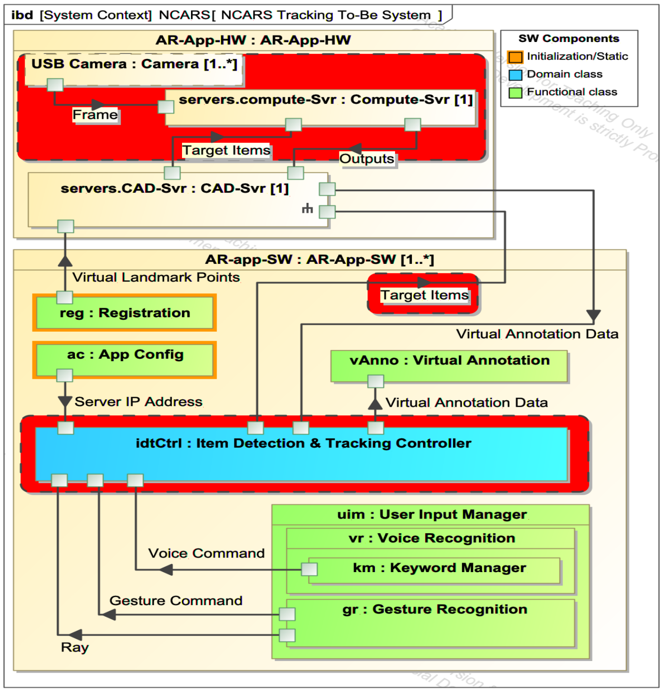

Fig. 11. Tracking to-be system

as long as the target item is found in the streaming data from the camera. This communication happens asynchronously; thus, the connection protocol, such as the WebSocket, is utilized with the controller being a client to maintain the communication with the server.

Preparing the system to detect and classify specific items in the physical space from a video capture device requires a number of tasks. Components required to undertake these tasks are summarized in Fig. 12. It includes the required software and hardware components that enable these tasks. The pre-trained You Only Look Once version 8 (YOLOv8) algorithm [21] is utilized for item classification, and it is deployed onto the computer server. Pegs and washers are examples of Item types of interest for the table-top conveyor system which serves as the test-bench for the model-based design of the AR applications.

With the structure and behavior of the system realized, a prototype of software components for the new capability is designed and tested with an AR device. The following summarizes the results of a partial potential solution for detecting items in motion and annotating them, which are tasks required for the new requirements.

A. *Demonstration of integration of new capability*

A new class of Item Detection & Tracking Controller class is coded and integrated into the existing AR application. The tracking functionality is designed to be enabled when an user sends out a voice command, i.e., "start tracking," within the Annotation app. The *to-be* system prototype is designed to detect items of interest, pegs and washers, at eight locations along the conveyor belt. Fig. 13 shows the result of

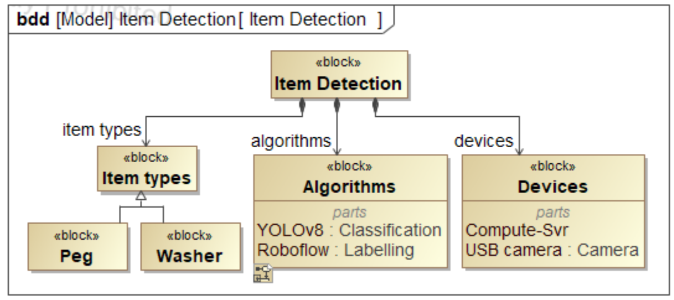

Fig. 12. Item detection components

YOLOv8 algorithm for real-time detection and classification using images streamed from an external camera. The camera is placed at a designated location; i.e., Fig. 1 is a view seen by the camera. The bounding box with a label p2 indicates that a peg is detected at position 2. The value next to the label is a confidence score indicating the probability that the predicted box contains the classified item, i.e., peg at position 2. Upon detection, the system initiates the transmission of pertinent information to the AR device in the form of an information bubble with a latency of about 1 second. Fig. 14 demonstrates the user's view from the AR device. This virtual annotation will move as the item in physical space changes its position.

Fig. 13. Camera view (zoom-in)

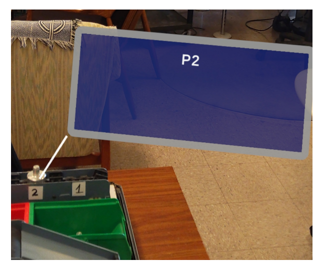

Fig. 14. AR user view (zoom-in)

The model-based representation of the AR *as-is* system enabled an agile deployment of a new capability that was required by a user, allowing iterative and incremental system updates, and a modular SysML design. A short demonstration of the *as-is* and *to-be* AR application is presented in [22].

VI. Discussion

A model-based representation of an existing AR application was utilized to integrate new software capability to satisfy a new user requirement. With a focus on the human-centric design of emerging technologies, MBSE and SysML are proposed as a means to engage system users, software engineers, and systems engineers on a shared platform to conceptualize and converge on new system requirements.

Whereas software engineering tools such as class diagrams allow visualization of the structure of the software design, they often include details that complicate the representation of the key architecture as the system becomes complex. They also have limitations in demonstrating the run-time connectivity of the functions involved. To extract a minimum set of software components that perform the required functions, software developers who designed NCARS worked with systems engineers in the co-design of the proposed architectural diagrams using SysML.

An existing model-based representation of the AR system that captured the high-level NCARS capability was first extended by incorporating the underlying software components that supports the system functions. Software architecture, their connectivity, and behavior are described by SysML block definition, internal block, and sequence diagrams, respectively. A modular design approach is implemented which supports iterative and agile development. The extension was demonstrated by adding a new capability to the NCARS Annotation function, enabling it to detect and annotate moving items in the physical space. A new class for Item Detection and Tracking was proposed, and its dependence on existing objects in NCARS and the requirement for new functions for the software developer to implement was specified using a SysML internal block diagram. A software update with the new specifications was carried out and verified to meet the new requirement.

In future work, the SysML platform will be accessible through a virtual collaborative space to enable multi-domain users, as discussed in [17], to propose new capabilities (e.g., integration of event-based cameras or new interaction gestures) and use cases. The result of these proposal will be explored using and integrated model-based systems and software engineering design. Additionally, a means to evaluate the performance of the proposed framework is developed. The integration of feedback from software and systems designers is essential to ensure the value of the framework to help facilitate the human-in-the-loop CPS design.

Acknowledgment

The authors acknowledge support of this research from the National Science Foundation Future of Work at the Human Technology Frontier program under Grant Nos. #2128749 and #2128950 and the National Science Foundation Innovations in Graduate Education in Cyber-Physical Systems Engineering under Grant No. #2105701. Any opinions, findings, conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and industry 5.0—inception, conception and perception," *Journal of Manufacturing Systems*, vol. 61, pp. 530–535, 2021, doi: 10.1016/j.jmsy.2021.10.006.

[2] D. Ivanov, "The industry 5.0 framework: Viability-based integration of the resilience, sustainability, and human-centricity perspectives," *International Journal of Production Research*, vol. 61, no. 5, pp. 1683–1695, 2023, doi: 10.1080/00207543.2022.2118892.

[3] J. Alves, T. M. Lima, and P. D. Gaspar, "Is industry 5.0 a human-centred approach? a systematic review," *Processes*, vol. 11, no. 1, p. 193, 2023, doi: 10.3390/pr11010193.

[4] A. Grech, J. Mehnen, and A. Wodehouse, "An extended ai-experience: Industry 5.0 in creative product innovation," *Sensors*, vol. 23, no. 6, p. 3009, 2023, doi: 10.3390/s23063009.

[5] M. Quandt, T. Beinke, and M. Freitag, "User-centered evaluation of an augmented reality-based assistance system for maintenance," *Procedia CIRP*, vol. 93, pp. 921–926, 2020, doi: 10.1016/j.procir.2020.03.053.

[6] V. Villani, L. Sabattini, P. Barańska, E. Callegati, J. N. Czerniak, A. Debbache, M. Fahimipirehgalin, A. Gallasch, F. Loch, R. Maida, A. Mertens, Z. Mockałło, F. Monica, V. Nitsch, E. Talas, E. Toschi, B. Vogel-Heuser, J. Willems, D. Żołnierczyk Zreda, and C. Fantuzzi, "The inclusive system: A general framework for adaptive industrial automation," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 1969–1982, 2021, doi: 10.1109/TASE.2020.3027876.

[7] S. Werrlich, C. Lorber, P.-A. Nguyen, C. E. F. Yanez, and G. Notni, "Assembly training: Comparing the effects of head-mounted displays and face-to-face training," in *Virtual, Augmented and Mixed Reality: Interaction, Navigation, Visualization, Embodiment, and Simulation*, ser. Lecture Notes in Computer Science, J. Y. Chen and G. Fragomeni, Eds. Springer International Publishing, Cham, 2018, vol. 10909, pp. 462–476, doi: 10.1007/978-3-319-91581-4_35.

[8] A. Malta, T. Farinha, and M. Mendes, "Augmented reality in maintenance—history and perspectives," *Journal of Imaging*, vol. 9, no. 7, p. 142, 2023, doi: 10.3390/jimaging9070142.

[9] W. Tao, Z.-H. Lai, M. C. Leu, Z. Yin, and R. Qin, "A self-aware and active-guiding training & assistant system for worker-centered intelligent manufacturing," *Manufacturing letters*, vol. 21, pp. 45–49, 2019, doi: 10.1016/j.mfglet.2019.08.003.

[10] H. G. Kenngott, A. A. Preukschas, M. Wagner, F. Nickel, M. Müller, N. Bellemann, C. Stock, M. Fangerau, B. Radeleff, H.-U. Kauczor, H.-P. Meinzer, L. Maier-Hein, and B. P. Müller-Stich, "Mobile, real-time, and point-of-care augmented reality is robust, accurate, and feasible: a prospective pilot study," *Surgical endoscopy*, vol. 32, pp. 2958–2967, 2018, doi: 10.1007/s00464-018-6151-y.

[11] J. Kim, S. Saguna, C. Åhlund, and K. Mitra, "Augmented reality-assisted healthcare system for caregivers in smart regions," in *2021 IEEE International Smart Cities Conference (ISC2)*, Manchester, United Kingdom, 2021, pp. 1–7, doi: 10.1109/ISC253183.2021.9562927.

[12] K. Klinker, M. Weische, and H. Krcmar, "Digital transformation in health care: Augmented reality for hands-free service innovation," *Information Systems Frontiers*, vol. 22, pp. 1419–1431, 2020, doi: 10.1007/s10796-019-09937-7.

[13] C. Moro, C. Phelps, P. Redmond, and Z. Stromberga, "Hololens and mobile augmented reality in medical and health science education: A randomised controlled trial," *British Journal of Educational Technology*, vol. 52, no. 2, pp. 680–694, 2021, doi: 10.1111/bjet.13049.

[14] National Academies of Sciences, Engineering, and Medicine, *A design thinking, systems approach to well-being within education and practice: Proceedings of workshop*. Washington, DC: The National Academies Press, 2019, doi: 10.17226/25151.

[15] M. C. Hause and F. Thom, "7.2.1 building bridges between systems and software with sysml and uml," *INCOSE International Symposium*, vol. 18, no. 1, pp. 797–811, 2008, doi: 10.1002/j.2334-5837.2008.tb00843.x.

[16] E. Aoki, B. Tran, V. Tran, K. Soth, C. Thompson, S. Tripathy, K. Chandra, and S. Sastry, "Representing augmented reality applications in systems modeling language," in *2023 IEEE International Systems Conference (SysCon)*, Vancouver, BC, Canada, 2023, pp. 1–8, doi: 10.1109/SysCon53073.2023.10131060.

[17] E. Aoki, B. Tran, N. E. Uhunsere, S. T. Tripathy, C. Thompson, S. Sastry, and K. Chandra, "Multidisciplinary authoring – a critical foundation for augmented reality systems in training and education," in *2023 ASEE Annual Conference & Exposition*, Baltimore, MD, USA, 2023. [Online]. Available: https://peer.asee.org/43690

[18] Unity Technologies, "Unity engine." [Online]. Available: https://unity.com/products/unity-engine

[19] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com

[20] B. Tran and S. Sastry, "Mapping a virtual view to the physical world to guide the completion of complex task sequences," in *2019 IEEE International Systems Conference (SysCon)*, Orlando, FL, USA, 2019, pp. 1–8, doi: 10.1109/SYSCON.2019.8836724.

[21] Ultralytics, "Yolov8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[22] Center for Advanced Computation & Telecommunications Channel, "NCARS Annotation and Tracking Features in the Augmented Reality Application: A Video Demonstration," (Feb 25, 2024). [Online]. Available: https://youtu.be/9EgqrzEAvJI