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, there is now significant opportunity for collaborative design among system users, software developers, and systems engineers. Model-based systems engineering and systems modeling languages enable 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 Annotation function, an existing MBSE model of the AR system is extended to include the structure and behavior of relevant software components. 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 that represent the integration of machine-learning algorithms for item classification and virtual annotations from the AR system.**

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

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 well-being can be characterized with respect to the resilience of humans in utilizing the technology to enhance their skills and collaborative capacity and in building 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. Quandt et al. [5] present a user-centered design approach for evaluating an AR-based assistance system. Villani et al. [6] present an adaptive industrial automation system that was evaluated considering real production tasks carried out by shop-floor workers.

These studies demonstrate the successful implementation of systems that effectively promote better human-technology interaction. Findings from all the different methodologies suggest that combining AI and VR in systems that highlight the importance of human-centered design can significantly enhance the product ideation process in engineering design and improve the system's usability and efficacy. Although some challenges are associated with user-technology interactions, the methodology presented by the references above shows 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 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 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 generally been applied to describe the requirements, structure, and behavior of complex systems with less attention given to its potential for describing the software that drives these systems. 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. In this paper, we demonstrate this advantage using a specific application in an AR system and discuss 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 that will allow a 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 the essential software dependencies and enable new capabilities to be designed efficiently. 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 specific table-top conveyor system 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 that allows them to efficiently write the code to support the required capability.

The AR application was developed for a conveyor system shown in Fig. 1 which is equipped with ferrous and nonferrous metal detector sensors to detect the presence of metal pegs and plastic washers. The conveyor system consists of continuous belt loops that move from one end to another, and sensors are placed at specific locations to detect the presence of the pegs and washers. The integration of AR devices into the conveyor system enhances the operation efficiency by providing the operator with real-time data of items. This AR augmentation provides seamless monitoring and management of the material in transit, allowing quick decisions and interventions when applicable.

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

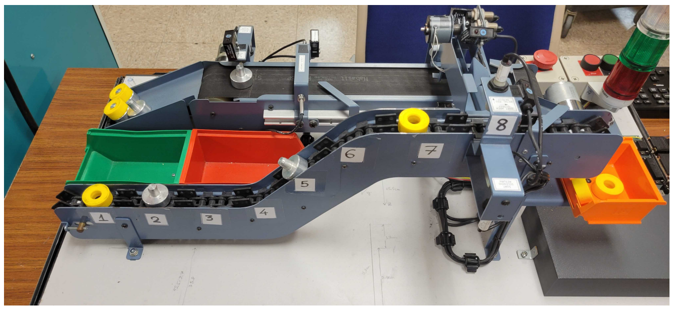


Fig. 1. Conveyor belt system

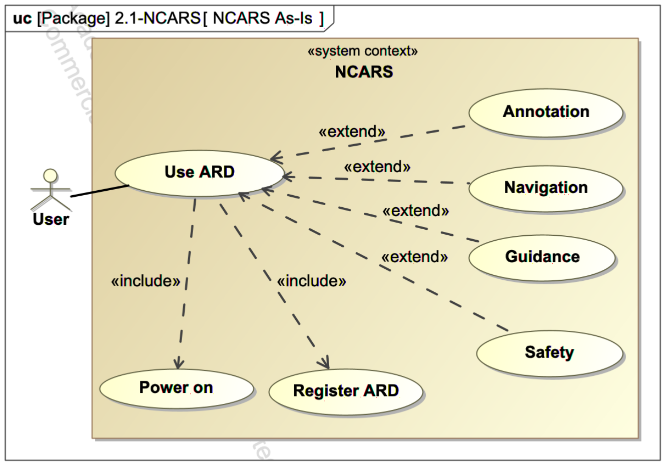
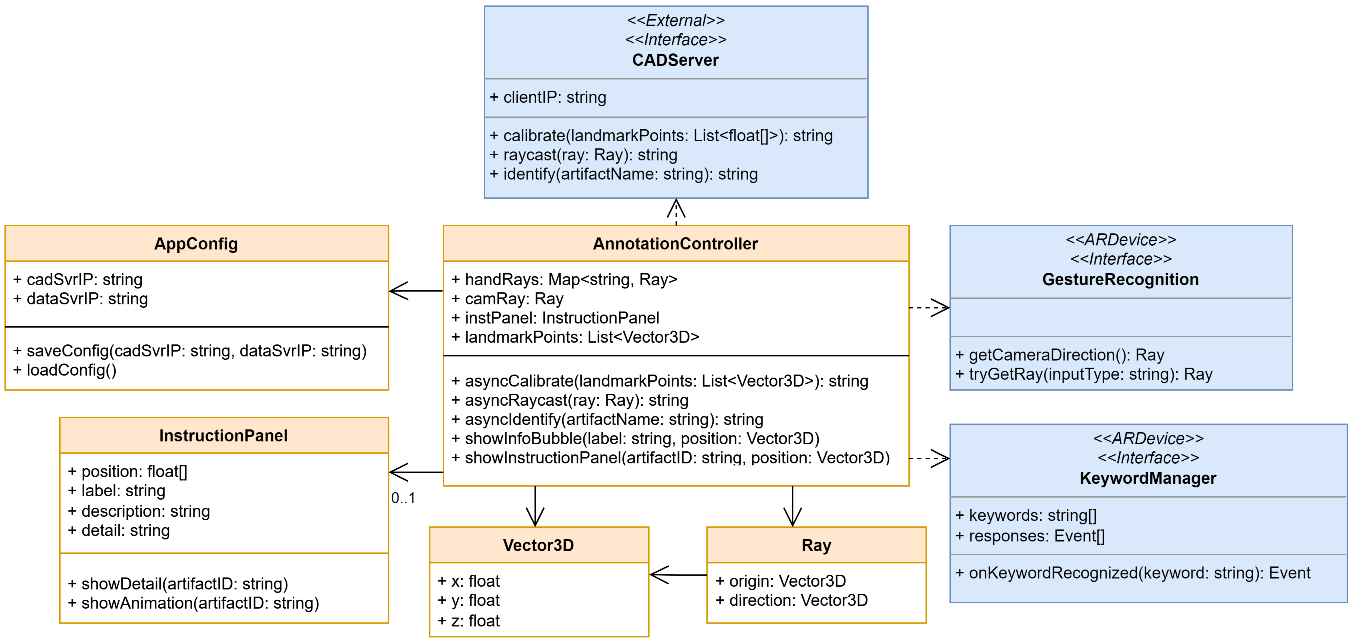


Fig. 2. Primary (As-Is) capabilities of NCARS

The NCARS framework uses a registration method presented in [18]. The registration method and NCARS architecture restrict the annotation of fixed locations on the conveyor system in the AR field of view (FOV). In this paper, we consider 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. This paper discusses the extension both from the perspective of the techniques used and from the perspective of the impact of the change on the SysML model for NCARS.

III. AR Application Software Structure and Dependencies

In this section, the software design of the Annotation function in NCARS is discussed. The Annotation function allows the system to present annotations in the FOV that have useful information relating to the user's point of interest. The



<<External>>
<<Interface>>
CADServer
+ clientIP: string
+ calibrate(landmarkPoints: List<float[]>): string
+ raycast(ray: Ray): string
+ identify(artifactName: string): string
AppConfig
+ cadSvrIP: string
+ dataSvrIP: string
+ saveConfig(cadSvrIP: string, dataSvrIP: string)
+ loadConfig()
AnnotationController
+ handRays: Map<string, Ray>
+ camRay: Ray
+ instPanel: InstructionPanel
+ landmarkPoints: List<Vector3D>
+ asyncCalibrate(landmarkPoints: List<Vector3D>): string
+ asyncRaycast(ray: Ray): string
+ asyncIdentify(artifactName: string): string
+ showInfoBubble(label: string, position: Vector3D)
+ showInstructionPanel(artifactID: string, position: Vector3D)
<<ARDevice>>
<<Interface>>
GestureRecognition
+ getCameraDirection(): Ray
+ tryGetRay(inputType: string): Ray
InstructionPanel
+ position: float[]
+ label: string
+ description: string
+ detail: string
+ showDetail(artifactID: string)
+ showAnimation(artifactID: string)
0..1
<<ARDevice>>
<<Interface>>
KeywordManager
+ keywords: string[]
+ responses: Event[]
+ onKeywordRecognized(keyword: string): Event
Vector3D
+ x: float
+ y: float
+ z: float
Ray
+ origin: Vector3D
+ direction: Vector3D

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

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 the NCARS framework 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. The class blocks represent the elements that are changeable by the software designer, while 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 of the design element are represented by the block's properties, and the behaviors are represented by the block's methods. In each block, the properties and methods are shown in $\langle$property: type$\rangle$ and $\langle$method(): return type$\rangle$ formats. The return type of a method is omitted if it does not return any element. The solid and dashed line arrows represent the reference and dependency relationships, respectively.

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 [18] 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.

The NCARS framework has been designed within the larger context of an application stack. A game engine such as Unity [19] or Unreal [20] is the bottommost layer and provides the rendering capability for the applications. It renders multimodel graphics and audio. The NCARS framework rides on the game engine and simplifies the use of the game engine for specific tasks — Annotation, Guidance, Navigation, and Safety. NCARS offers a more restrictive interaction than a general game but makes it easier for application development. On top of NCARS is the Human-Centric 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. 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.

IV. Extending 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 AR application. The AR hardware includes devices, such as the Microsoft® HoloLens and Magic Leap®, and various servers that support 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 show new components added to the as-is system to extend its capability. The use of these new components are 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 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 and Data, CAD, and Compute servers. If child blocks can exist independently of the parent, a directed aggregation shown in a solid line with an open diamond is used. Hence, NCARS utilizes physical systems and networks, and AR-App-HW uses AR devices and USB Cameras. A solid line with an open arrowhead represents a unidirectional association with another block, which shows AR system users' interactions with the NCARS system for particular purposes, noted against the arrow. The dotted line with the open arrow shows a type of dependency between two blocks. *use* indicates that a block uses another whereas *Item Flow* specifies items being sent from one element to

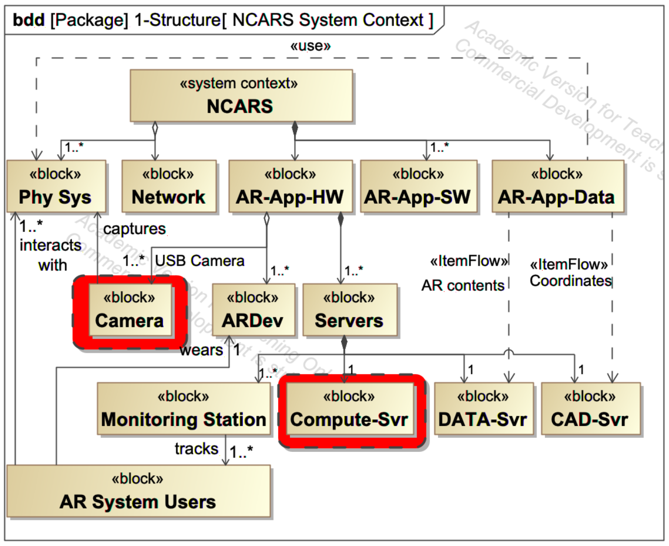


Fig. 4. High level NCARS system structure

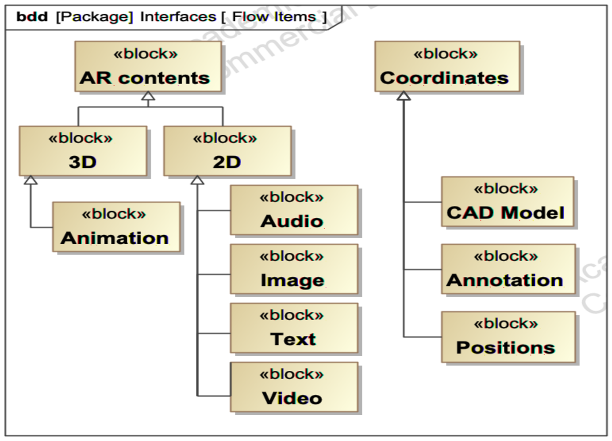


Fig. 5. Components saved in DATA and CAD servers from Data sources

the other. Fig. 5 describes data types stored in Data and CAD servers. These data may correspond to the contents from domain experts as described in Section III. A solid line with a closed arrowhead shows the generalization relationship, e.g., Animation is the sub-type of 3D AR content.

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 system for the four tasks specified to visualize reusable and specific components. 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.

Note that the *GestureRecognition* and *KeywordManager* interface classes in Fig. 3 belong to the *User Input Manager*

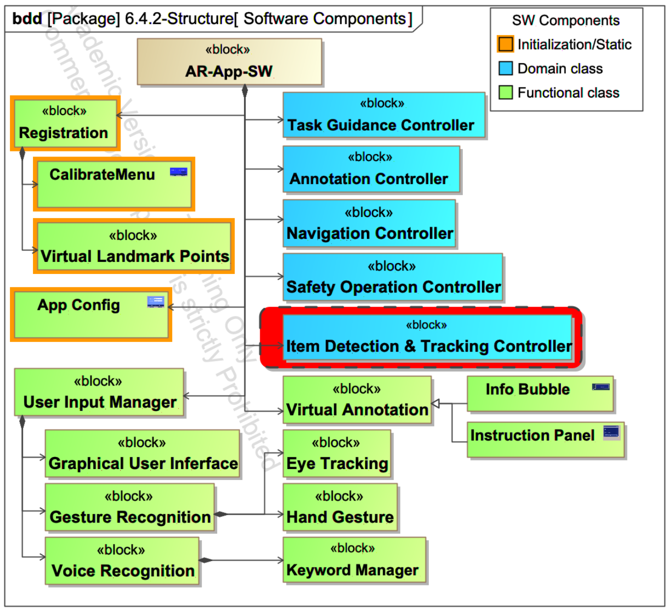

bdd [Package] 6.4.2-Structure[Software Components]
SW Components
Initialization/Static
Domain class
Functional class
«block» AR-App-SW
«block» Registration
CalibrateMenu
«block» Virtual Landmark Points
App Config
«block» Task Guidance Controller
«block» Annotation Controller
«block» Navigation Controller
«block» Safety Operation Controller
«block» Item Detection & Tracking Controller
«block» User Input Manager
«block» Graphical User Inferface
«block» Gesture Recognition
«block» Voice Recognition
«block» Virtual Annotation
Info Bubble
Instruction Panel
«block» Eye Tracking
«block» Hand Gesture
«block» Keyword Manager

Fig. 6. High-level AR application software structure

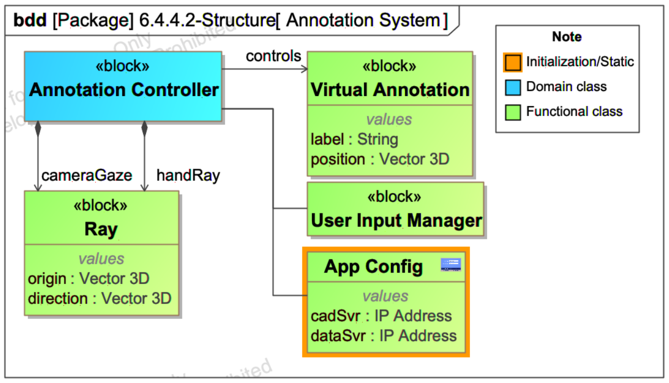


Fig. 7. Annotation as-is system structure

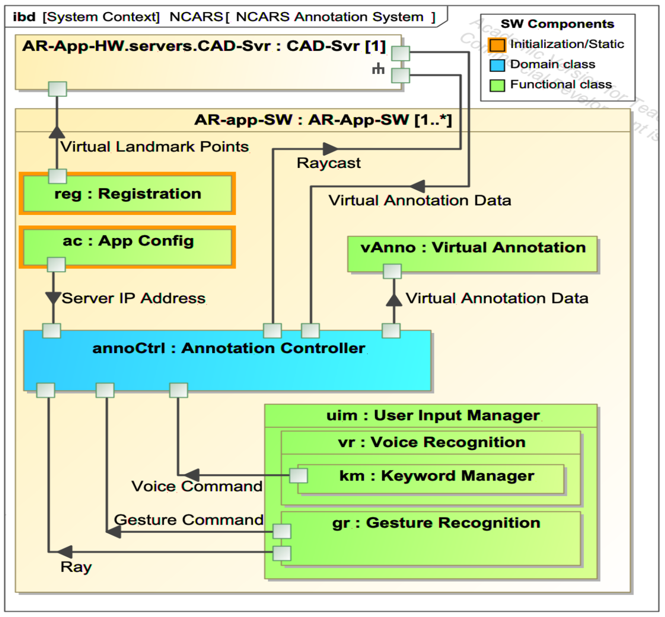


Fig. 8. Annotation as-is system connection

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 the NCARS system.

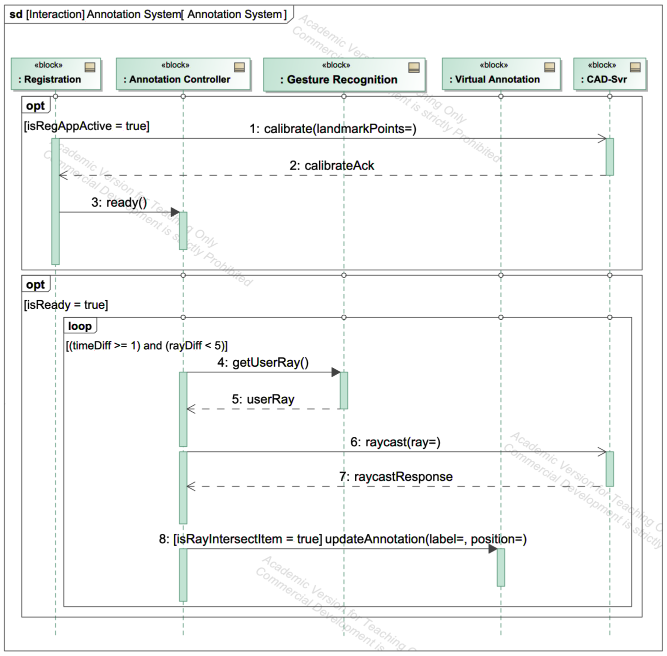
The orange-colored boundaries indicate that the object instances corresponding to these classes would be executed once for the application. In contrast, object instances corresponding to classes without the orange highlights have components that are executed multiple times until the object instance is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

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

With the high-level structure constructed, each of NCARS functions can be further described. Fig. 7 shows the Annotation system, that is before a new capability is integrated. 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 also 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 controls the Virtual Annotation.

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.

The sequence diagram shown in Fig. 9 provides further detail on the dynamic behavior of the components described in Fig. 8 as particular operations are invoked sequentially. An open arrow indicates asynchronous operation whereas a closed arrow means synchronous behavior. A dashed line represents a response. 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



sd [Interaction] Annotation System[Annotation System]
«block» : Registration
«block» : Annotation Controller
«block» : Gesture Recognition
«block» : Virtual Annotation
«block» : CAD-Svr
opt
[isRegAppActive = true]
1: calibrate(landmarkPoints=)
2: calibrateAck
3: ready()
opt
[isReady = true]
loop
[(timeDiff >= 1) and (rayDiff < 5)]
4: getUserRay()
5: userRay
6: raycast(ray=)
7: raycastResponse
8: [isRayIntersectItem = true] updateAnnotation(label=, position=)

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

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 system 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 every 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.

Analyzing the structure of the as-is software architecture described helps inform where a new capability can be added and also what components would be affected or reusable. The next section further describes how additional components are designed to achieve an extension of the as-is 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. 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.

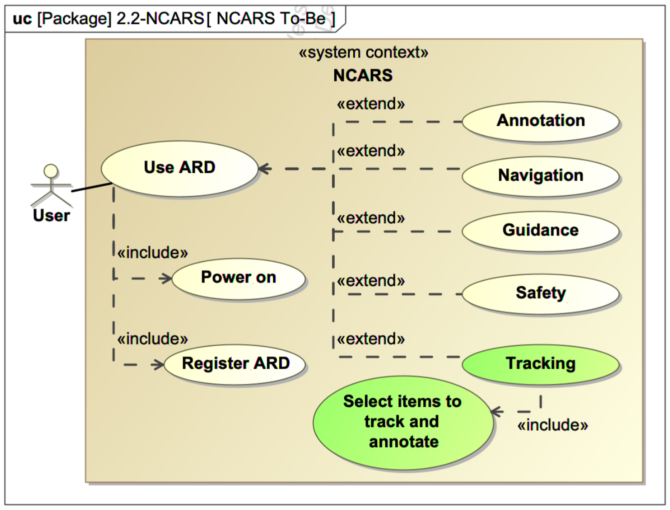


Fig. 10. Use Case describing a new capability of NCARS

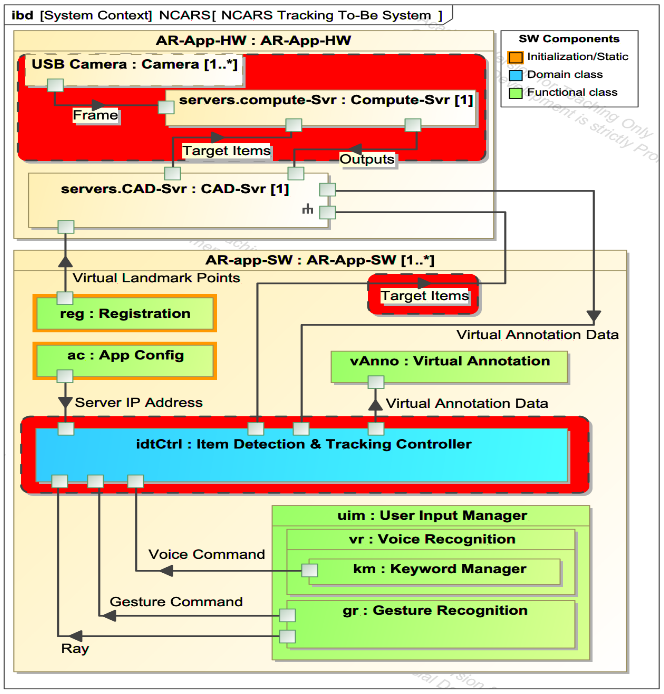


Fig. 11. Tracking to-be system

Note that the to-be system shown in Fig. 11 has a similar structure as the Annotation as-is system described in Fig. 8. New components surrounded by the red rectangles are added to achieve the new capability. The Item Detection & Tracking Controller receives a user's items of interest and sends them 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 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. The tasks undertaken to detect and classify these items are shown in Fig. 12. It includes the required software and hardware components that enable these tasks. 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 AR applications.

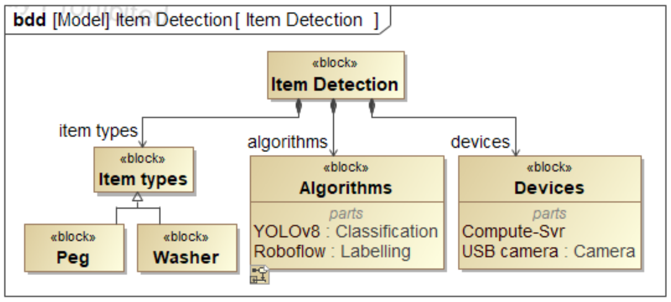


Fig. 12. Item detection components

The state machine diagram shown in Fig. 13 outlines the process of creating a training and tracking application. The You Only Look Once version 8 (YOLOv8) algorithm [21] is utilized. The state machine depicts the sequential events, such as a collection of image and video data from a camera system, followed by pre-processing tasks of labeling the data with bounding boxes and creating train, test, and validation data sets. Further, the training data is scaled with various transformations to improve the robustness of the classification algorithm. The final step is the training using a neural network classifier and iterating the process until a favorable accuracy is achieved with the model. Finally, the model is deployed onto the compute server.

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 part of the new requirements.

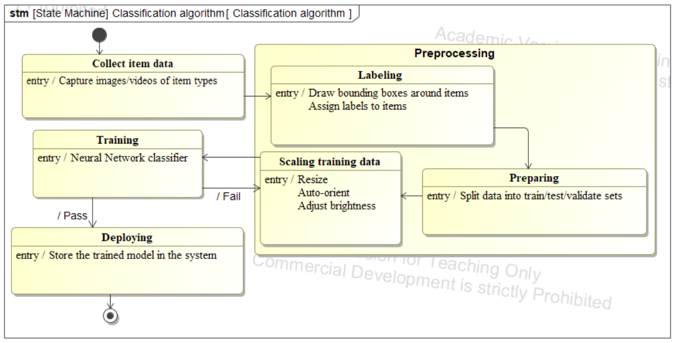

stm [State Machine] Classification algorithm[Classification algorithm]
Collect item data
entry / Capture images/videos of item types
Preprocessing
Labeling
entry / Draw bounding boxes around items
Assign labels to items
Training
entry / Neural Network classifier
Scaling training data
entry / Resize
Auto-orient
Adjust brightness
Preparing
entry / Split data into train/test/validate sets
/ Fail
/ Pass
Deploying
entry / Store the trained model in the system
Academic Version for Teaching Only
Commercial Development is strictly Prohibited

Fig. 13. Detection and Tracking algorithm preparation

A. Demonstration of integration of new capability

The to-be system prototype is designed to detect items of interest, pegs and washers, at eight particular locations. Fig. 14 shows the result of YOLOv8 algorithm for real-time detection and classification using images streamed from an external camera. This camera is placed where it captures the beginning of the tabletop conveyor belt. The bounding box with a label p2 indicates that a peg is detected at position 2, and this data is sent to the AR device. 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. 15 demonstrates the user's view from the AR device. This virtual annotation will move as the item in physical space changes its position.

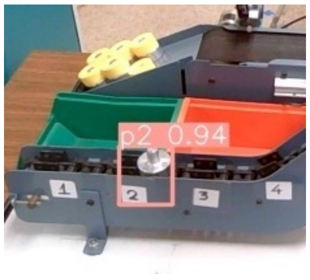


Fig. 14. Camera view (zoom-in)

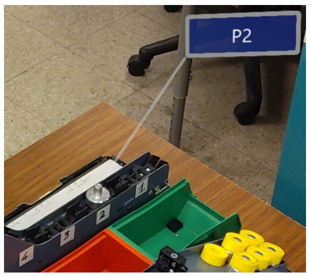


Fig. 15. AR user view (zoom-in)

The model-based representation of the AR as-is system enabled an agile deployment of a new capability required by a user, with support for iterative and incremental system updates, embracing a modular SysML approach. The next phase of the research involves evaluating and refining the prototype against new requirements and engaging software and system engineers in concurrent the model-based design of the system and its software.

VI. Discussion

This paper addressed the integration of new software capability for an existing AR application, utilizing a model-based representation of the system. With a focus on the human-centric design of emerging technologies, MBSE and SysML are proposed as a means to engage system users and software and system engineers on a shared platform to conceptualize and converge on new AR system requirements.

Whereas software engineering tools such as dependency graphs allow visualization of the structure of the software design, they often include details that complicate the representation of the key architecture. 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 the NCARS framework 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 AR capability at a high level was first extended by incorporating the underlying software architecture that supports the system functions. The extension was demonstrated by adding a new capability to the Annotation function of NCARS, enabling it to track 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. The prototype system implemented the new structure, and its work was verified. In future work, the SysML platform will be accessible through a virtual collaborative space that allows multi-domain users to propose new capabilities and use cases and envision the result of their proposal using model-based systems and software engineering.

As future work, it may be beneficial if such applications offer a new capability to promote robust communications or feedback system between the practical and supervised system users. It is worth exploring that articulating the model by integrating software and systems engineering help facilitate the human-in-the-loop CPS design with multiple disciplines.

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] 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.

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

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

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