From Flat Step Lists to Adaptive Manual Assembly Instructions: A Dynamic Programming Approach to Instruction Chunking

André Dirks[1] and Thies Pfeiffer[1]

University of Applied Sciences Emden/Leer, Emden, Germany
{andre.dirks,thies.pfeiffer}@hs-emden-leer.de

Abstract. Within the Industry 4.0/5.0 paradigms, manual assembly is becoming increasingly complex, making digital assistance systems crucial for guiding operators. However, as operators become more experienced, they may become accustomed to highly granular and static instructions, causing them to ignore the assistance and thus might miss critical information (safety, quality, variations). Therefore, to remain relevant, assistance systems must dynamically adapt their information density to the user. To address this challenge, we propose a chunking engine (where chunking refers to the grouping of sequential steps into meaningful instructional units) that enriches atomic assembly steps with complexity and context attributes, using a dynamic programming algorithm to generate heuristically balanced instructional chunks. The paper concludes with a computational demonstration showing that the engine reduces 23 atomic steps into balanced instructional chunks, adapting the output to both novice and expert operator profiles.

Keywords: Cognitive Load · Human-Computer Interaction · Adaptive User Interfaces · Assembly · Hierarchical Task Analysis · Industry 5.0.

1 Introduction

In manual manufacturing, static paper-based instructions are increasingly being replaced by digital assistance systems. As the industry transitions toward the Industry 5.0 paradigm, the focus shifts to empowering the human worker within highly connected, automated environments. Building on the Operator 4.0 concept [11], Thorvald et al. [14] describe the "Cognitive Operator," fundamentally shifting the perspective of the modern worker from manual labor to knowledge and reasoning capabilities. Cognitive Assistance Systems (CAS) pivot from mere information delivery to actively supporting the operator's cognitive information processing [10] as the operator faces growing complexity due to mass customization.

Despite these advancements, a practical information mismatch persists. Johansson et al. [6] interviewed operators across truck manufacturing plants regarding their digital work instructions. They found that 83% of operators in the

most complex environments only use instructions to look up highly specific information. One operator summarized the issue by stating that current instructions are "too extensive for experienced operators and too poor for novices."

To address this, CAS designers must move beyond static delivery and embrace interfaces that support chunking dynamically. We propose an algorithmic chunking engine that groups granular assembly steps into adaptive instructional units. Assuming an underlying process planner outputs a flat sequence of tasks at the most granular level, these atomic steps are first enriched with physical complexity scores, semantic intent classifications, and context metadata. Our engine then uses dynamic programming to optimize over these variables and aggregate the operations into cohesive cognitive chunks from the bottom up. By isolating this semantic chunking layer, this proof-of-concept framework demonstrates how instructions can be mathematically tailored to an operator's specific expertise level without requiring manual authoring for every proficiency tier.

2 Related Work

2.1 Cognitive Constraints in Assembly Tasks

While the Johansson study identifies the symptoms of poor instruction design, the cause lies in the biological constraints of the human brain. The primary bottleneck is working memory, which has a fixed capacity that is rapidly exhausted in stimulus-heavy environments. Cowan [1] established this capacity at approximately four discrete chunks of information. When an instruction set exceeds this limit, cognitive overflow is likely. Sweller introduced the concept of element interactivity as a predictor of task difficulty [13]. Element interactivity is the degree to which elements of a task must be processed simultaneously to be understood. For instance, tightening a screw requires simultaneously knowing the correct tool, the target torque, and the spatial orientation of the component. Crucially, the level of operator expertise directly influences perceived element interactivity: as familiarity grows, previously distinct elements merge into single cognitive units. In the context of manual assembly, providing overly detailed instructions for experts results in an increase in extraneous cognitive load. This may lead to the Expertise Reversal Effect [7]: as a worker becomes proficient, the element interactivity of a task decreases because they chunk multiple steps into a single mental schema. To an expert, “Take, position, and fasten bolt” is one chunk; to a novice, it is three. When a CAS forces an expert to process these as separate steps, it disregards their existing mental models and risks being ignored entirely, which becomes a concern when product mass-customization requires the worker to remain alert to non-routine changes.

2.2 Structuring Work Instructions

A solution to this problem is the dynamic adaptation of how an assembly sequence is instructed. By chunking the instructions for several steps into one

unit according to the operator's level of expertise, the system can deliver the right level of detail to the right person. To build a dynamic chunking system, the underlying physical and cognitive demands of the assembly process must be structurally and mathematically understood. Kelm et al. [8] created a set of syntax modules for assembly instructions based on instruction-relevant relationships, such as supply locations, quantities, or required production equipment. However, these operational factors are often not represented in a standard Hierarchical Task Analysis (HTA), which primarily groups parts logically (i.e., all parts of the engine block make up the engine block). HTA is a widely used tool to inform work instructions [12]. Still, HTA structures fail to portray the difficulty or complexity of any given atomic step. There have been efforts to objectively calculate the complexity of manufacturing operations to fill this gap. ElMaraghy et al. [3] created the "Operational Complexity Index" as a standardized metric. This index mathematically quantifies the physical and cognitive demands of a task by integrating the total volume of operational steps with a diversity ratio of contextual changes (e.g., varying tools or locations) and a relative complexity coefficient representing individual step difficulty. An alternative approach to determining task difficulty are the Basic Assembly Complexity (CXB) criteria defined by Falck et al. [4], which offer 16 objective physical problem areas (e.g., visual occlusion, lack of feedback, tight tolerances) that drive quality errors. As established by Mattsson et al. [9], physical constraints dictate the operator's Perceived Production Complexity, translating geometric difficulties into direct cognitive penalties. These complexity metrics provide the empirical inspiration for the step-level scoring used in the chunking engine proposed in the next section.

3 Algorithm Design

3.1 Anticipated Challenges in Algorithmic Chunking

Chunking atomic steps of any process into adaptive instructional units is a non-trivial task, because summarization or abstraction introduces new structural risks. A naive system might summarize a five-step process into one dense chunk of four steps and one fragmented chunk of a single step. This imbalance may break the flow of the operator. If the system applies incorrect semantic boundaries, it risks phase blurring: an algorithm may merge the final step of one logical assembly with the initial step of a subsequent assembly, e.g. when the immediate spatial context remains largely the same. Hence, the mathematical rules need to produce consistent results, even for edge cases. When adapting for complexity by limiting chunks with certain heuristics, the system may cause hyper-fragmentation (i.e., producing chunks that consist of single operations). Additionally, an algorithm may group steps that belong together conceptually but should remain separated due to spatial attributes, such as needing to grab a tool before changing position at the station.

3.2 Quantifying Task Complexity

Before adapting instructions to the user, we must first establish what makes any given atomic step difficult. We propose a base load of 1 for any given atomic step, representing the minimum cognitive cost of initiating any action. Additional complexity attributes capture the specific difficulty of the operation and are added to this base load.

3.3 Adapting to the User

Adding adaptivity to the system requires modeling operator expertise at two levels. The first is a global expertise level $x_w \in [0, 1]$, which represents the operator's overall proficiency and scales the perceived complexity of every step in the sequence. Following the ILUO methodology, we map novice operators to $x_w = 1.0$ and expert operators to $x_w = 0.25$, with intermediate levels in between. In a production deployment, this value would be supplied automatically by a skills matrix or Worker AAS.

The second level accounts for the fact that the impact of expertise is not uniform across all operations. "Tighten these screws" may be done automatically by an expert, while a novice still needs specific instructions. However, "Take the screwdriver" requires little cognitive effort regardless of expertise level. We therefore introduce a per-step expertise sensitivity parameter $\alpha(s_i) \geq 0$ to model this non-linear relationship, allowing individual steps to be more or less responsive to the global expertise scalar.

3.4 Context Matters

The physical context of each operation has a strong influence on how efficiently it can be carried out. To capture this, we define three context metadata variables for each step: `tool_required`, `component_type`, and `location_zone`. The dynamic programming engine checks whether these variables change between consecutive steps and assigns friction penalties accordingly. These penalties are not simple binary flags. Instead, the system applies context-aware rules that reflect real-world assembly behavior. For example, when either of two adjacent steps has a `MANIPULATE` intent, the engine suppresses both location and component penalties entirely, because physical manipulation spans across zones and parts.

A special case of this physical relationship exists between components and their associated fasteners. Because a screw does not exist independently but belongs to the component it secures, we argue that any transition penalty between a fastener and its primary component should be nullified. We call this principle *relational affordance*. Furthermore, the system infers physical tool-swapping friction through sequential proximity. Transitioning between an empty-hand state and a tool incurs baseline friction ($w = 1.0$), whereas swapping between two distinct hand-held tools incurs double friction ($w = 2.0$). This allows the algorithm to separate physically disjointed workflows without over-penalizing simple tool pick-ups.

3.5 Semantic Intent and Transition Penalties

While the physical complexity of a step establishes its base load, the cognitive friction experienced by an operator depends heavily on the semantic relationship between sequential actions. We classify each step's intent using DIN 8580 [2] for core manufacturing processes and VDI 2860 [15][1] for handling and controlling operations, and model the pairwise transition costs as a matrix. The chunking engine evaluates the semantic shift between two consecutive steps in order to identify low-friction workflows. For example, `MANIPULATE` followed by `ADD` represents a logical sequence, so the transition penalty is zero. Conversely, interrupting an action phase to handle a new object (`ADD` $\rightarrow$ `MANIPULATE`) disrupts the operator's mental model and incurs a penalty of 1.0. Shifting from a physical action to quality checking (e.g., `ADD` $\rightarrow$ `INSPECT`) introduces minor cognitive friction (0.3). In contrast, transitioning out of an inspection incurs zero penalty, as it represents the anticipated resumption of the workflow after quality has been confirmed. Critically, this matrix also enforces high-level task boundaries by assigning an infinite transition penalty when two steps represent opposing DIN 8580 operations, such as `SUBTRACT` (disassembly) and `ADD` (assembly). Subordinate steps like "take fasteners" or "take tool X" are classified as `MANIPULATE` and do not represent independent cognitive events. In this case, the zero transition cost ensures the DP engine groups these preparatory steps together with the primary action that follows them. While the matrix is designed to generalize across assembly scenarios, the specific penalty values should be treated as potential recalibration targets.

From \To	**ADD**	**SUBTRACT**	**MANIPULATE**	**INSPECT**
ADD	0.0	∞	1.0	0.3
SUBTRACT	∞	0.0	1.0	0.3
MANIPULATE	0.0	0.0	0.0	0.0
INSPECT	0.0	0.0	0.0	0.0

Table 1. Cognitive transition penalties $T(s_{i-1}, s_i)$ between sequential semantic intents. Note the directional asymmetry: moving from preparation to action (`MANIPULATE` $\rightarrow$ `ADD`) is frictionless (0.0), while interrupting an action phase to prepare a new tool (`ADD` $\rightarrow$ `MANIPULATE`) incurs a penalty (1.0).

3.6 Finding the Optimal Instructional Chunks

With step-level complexity and transition penalties defined, the chunking problem reduces to finding the best way to group a sequence of steps into balanced

[1] While formally withdrawn in 2018, the VDI 2860 taxonomy remains the foundational academic framework for classifying handling operations in assembly research.

instructional units. We treat the cognitive ceiling θ as a soft target rather than a hard limit, so the algorithm strongly discourages exceeding it while still being able to handle steps with high base complexity. This problem is solved using dynamic programming, which evaluates all possible groupings and selects the partition that keeps chunk loads as consistent as possible.

3.7 General Formulation

The components established across the preceding sections can be unified into a single expression. For any chunk c_{ab} consisting of a sequence of atomic steps from s_a to s_b, the total cognitive load $L(c_{ab}, w)$ perceived by a specific worker w is:

$$L(c_{ab}, w) = \sum_{i=a}^{b} \left((1 + C(s_i)) \cdot x_w^{\alpha(s_i)} \right) + \sum_{i=a+1}^{b} T(s_{i-1}, s_i) \tag{1}$$

Where $(1 + C(s_i))$ is the base complexity score of the step, combining the load of initiating any action (1) with the normalized physical criteria of the task ($C(s_i)$). $x_w \in [0, 1]$ is the worker's expertise level. $\alpha(s_i) \geq 0$ is the expertise sensitivity of the step s_i. When $\alpha(s_i) = 0$, the step costs the same regardless of who performs it. When $\alpha(s_i) = 1$, difficulty scales proportionally with the worker's novice level. Values of $\alpha(s_i) > 1$ can model tasks where veteran operators experience greater cognitive discounts, such as highly routine operations. For the specific implementation evaluated in this study, $\alpha(s_i)$ is held steady, effectively reducing $x_w^{\alpha(s_i)}$ term to a simple global scalar x_w. Finally, $T(s_{i-1}, s_i)$ represents the context-aware transition penalty incurred when moving from one step to the next. This term encapsulates the DIN 8580 semantic intent matrix, the relational affordances for fasteners, the tiered tool friction, and the spatial boundaries described above.

4 Proof of Concept

4.1 Results

To evaluate the chunking engine, we assess its output against the three structural risks identified in the section "Algorithm Design": load imbalance, phase blurring, and structural integrity, which encompasses both hyper-fragmentation and the absorption of tool-fetch steps. The test case is an existing HTA by Gavish et al. [5], comprising 23 atomic steps across two branches (Disassembly and Assembly), enriched with DIN 8580/VDI 2860 intent classifications and a subset of the Falck CXB criteria [4] that can be evaluated from a standard task description. We then examine how the system scales with expertise, including a complexity sensitivity experiment that compares two execution variants: one utilizing baseline complexity scores and another with elevated scores. See Table 2 for a summary of the results.

Table 2. Cognitive Load and Chunking Results by HTA Complexity and Expertise Level

HTA File	Expertise Level	Total Chunks	Average Load	Max Load
Normal	1.00 (Novice)	10	3.44	4.63
	0.75 (Intermediate)	9	3.94	5.08
	0.50 (Advanced)	7	4.04	4.61
	0.25 (Expert)	6	3.50	4.47
Complex	1.00 (Novice)	12	3.62	5.30
	0.75 (Intermediate)	11	3.86	5.22
	0.50 (Advanced)	9	3.93	4.80
	0.25 (Expert)	7	3.59	4.10

Load Balance. The novice profile produced 10 chunks with accumulated loads ranging from 2.0 to 4.633, centered around the $\theta = 4.0$ target. The DP optimization reduced load variance across the sequence, avoiding the drastic imbalances a naive grouping strategy would produce.

Structural Integrity. No single-step chunks were produced in either profile, as the 23 atomic steps consolidated into 10 chunks for the novice and 6 for the expert. The intent transition matrix grouped and separated subordinate preparatory steps around primary physical actions. As shown in Table 3, the novice profile treats tool-fetching as a discrete cognitive boundary, isolating the "Take pliers" step into Chunk 2. For the expert, the system leverages the load discount to pack the physical subtraction logic and the subsequent tool-pickup into a single cognitive schema (Expert Chunk 1). Furthermore, the engine utilizes intent masking to bridge actions. Expert Chunk 2 groups "Cut with the pliers" and "Remove board from support plate" because the transition occurs through a `MANIPULATE` step ("Take a screwdriver"), suppressing structural penalties and allowing the algorithm to merge separate physical tasks safely.

Phase Blurring. The Disassembly and Assembly branches were preserved as distinct macro-goals in both profiles. The infinite transition penalty assigned to opposing intents (`SUBTRACT` $\rightarrow$ `ADD`) acted as an absolute mathematical boundary, and the DP engine did not merge the final disassembly step with the initial assembly step. This indicates that context-aware transition penalties can enforce high-level structural logic without requiring manual top-down segmentation.

Expertise Scaling. The expertise adaptation is visible across both disassembly and assembly phases (Table 3). For the novice, `CHUNK_1` contains 3 steps at a load of 3.11. For the expert, `CHUNK_1` absorbs 5 disassembly steps into a single instructional unit at a load of 3.28, remaining within the cognitive ceiling. This pattern repeats during the assembly phase (`ADD`), where the novice profile yields 5 distinct groups (Chunks 6-10). The engine compresses this into 3 groups for the expert (Chunks 4-6). Expert Chunk 4 aggregates the entire subroutine of installing the board and securing the support plate into one cohesive unit. This reflects the expertise scalar ($x = 0.25$) reducing the effective per-step load, enabling the engine to group operations that a novice would need presented se-

quentially. See Table 3 for a direct comparison of produced chunks for novice and expert level of expertise.

Table 3. Chunk Boundary Comparison: Novice vs. Expert Profiles (Target Ceiling $\theta = 4.0$)

Novice Chunk	Intent	Atomic Step (Task Sequence)	Expert Chunk
C1 (3.11)	[SUB]	Remove the fixing screws	**C1** (3.28)
	[SUB]	Remove the actuator cover...	
	[SUB]	Remove the clamp from the stem	
C2 (2.00)	[MAN]	Take gripping pliers	
	[SUB]	Disconnect the wires with the pliers	
C3 (2.22)	[MAN]	Take cutting pliers	**C2** (4.06)
	[SUB]	Cut with the pliers	
C4 (4.00)	[MAN]	Take a screwdriver	
	[SUB]	Remove board from support plate...	
	[SUB]	Unscrew the fixing screws	**C3** (3.38)
C5 (3.41)	[SUB]	Remove support plate from structure	
	[SUB]	Remove screw holding the wires...	
C6 (3.33)	[ADD]	Substitute the electronic board	**C4** (4.47)
	[ADD]	Prepare the electronic support plate	
C7 (4.63)	[ADD]	Reconnect the electronic board wires	
	[ADD]	Fix support plate back onto structure	
C8 (3.74)	[ADD]	Snap the electronic board back...	**C5** (2.44)
	[ADD]	Reconnect the electronic board wiring	
C9 (4.63)	[ADD]	Keep the wires tidy with a nylon clip	
	[ADD]	Reconnect motor to electronic card	
	[ADD]	Replace the actuator cover	**C6** (3.41)
C10 (3.30)	[ADD]	Fix into place with the relative screws	
	[ADD]	Reassemble the clamp	

Sensitivity to Step Complexity. To evaluate whether the engine responds to physical task difficulty rather than operating solely on step count and transitions, we ran the same 23-step process with elevated Falck complexity scores across all steps. The normal variant uses baseline physical criteria (C_i ranging from 0.0 to 0.33), while the complex variant raises these values to reflect more demanding geometric and positional conditions. For the novice profile, the complex variant produced 12 chunks compared to 10, with an average load of 3.62 and a peak of 5.30. The peak load of 5.30 in the complex novice profile reflects the soft ceiling mechanism, which permits minor violations when steps are individually indivisible.

5 Conclusion

5.1 Summary

This paper presented a chunking engine that transforms a flat sequence of assembly steps into instructional units using dynamic programming. Each step is enriched with complexity scores, context metadata, and intent classifications, which the DP engine uses to partition the sequence into heuristically balanced chunks calibrated to the operator's expertise level.

The engine was evaluated on an existing HTA of an industrial control unit with 23 atomic steps. Using baseline complexity scores, the system produced 10 chunks for the novice profile, with no observed instances of load imbalance, hyper-fragmentation, or phase boundary violations. For the expert profile, the system produced 6 chunks, reflecting the expertise scalar reducing effective per-step load. When evaluated with elevated physical complexity scores, the engine produced finer-grained chunks (12 for novices, 7 for experts), confirming its sensitivity to underlying task difficulty without requiring altered transition rules.

5.2 Future Research and Limitations

This evaluation serves as a computational proof of concept, demonstrating that the engine adheres to its defined mathematical rules. However, because the $\theta = 4.0$ ceiling and the transition penalties are theoretical heuristics, future human-in-the-loop studies are required to validate whether these boundaries translate to measurable reductions in cognitive load for real operators. Furthermore, the evaluation covered a single 23-step assembly; testing on a wider range of HTAs with varying structural complexity is needed to generalize the findings. Future iterations should explore per-step expertise sensitivity values (α_i), rather than the uniform setting used here, to capture the non-linear relationship between expertise and task difficulty. Additionally, different experts may prefer different rule configurations. For instance, reducing the emphasis on tool changes for operators who are highly familiar with a given assembly. Finally, the manual enrichment of steps with complexity scores and intent classifications is time-consuming and heuristics are required to derive them automatically, e.g. with the help of machine learning.

Acknowledgments. This research was supported by the European Union - NextGenerationEU under the project TwinMaP (13IK028K), which operates as part of the overarching ARTWORK project framework.

Disclosure of Interests. The authors have no competing interests to declare.

References

1. Cowan, N.: The magical number 4 in short-term memory: A reconsideration of mental storage capacity. The Behavioral and brain sciences **24**, 87–114; discussion 114 (Mar 2001). https://doi.org/10.1017/S0140525X01003922

2. DIN 8580: Manufacturing processes – Terms and definitions, division (Sep 2003)
3. ElMaraghy, W., Urbanic, R.: Assessment of Manufacturing Operational Complexity. CIRP Annals **53**(1), 401–406 (2004). https://doi.org/10.1016/S0007-8506(07)60726-4
4. Falck, A.C., Örtengren, R., Rosenqvist, M., Söderberg, R.: Criteria for Assessment of Basic Manual Assembly Complexity **44**, 424–428. https://doi.org/10.1016/j.procir.2016.02.152, https://linkinghub.elsevier.com/retrieve/pii/S2212827116004340
5. Gavish, N., Krupenia, S., Gopher, D.: Task analysis for developing maintenance and assembly vr training simulators. Ergonomics in Design **21**(1), 13–18 (2013)
6. Johansson, P., Eriksson, G., Johansson, P., Malmsköld, L., Fast-Berglund, Å., Moestam, L.: ASSESSMENT BASED INFORMATION NEEDS IN MANUAL ASSEMBLY. DEStech Transactions on Engineering and Technology Research (icpr) (Mar 2018). https://doi.org/10.12783/dtetr/icpr2017/17637
7. Kalyuga, S., Ayres, P., Chandler, P., Sweller, J.: The expertise reversal effect. Educational Psychologist **38**(1), 23–31 (2003). https://doi.org/10.1207/S15326985EP3801_4
8. Kelm, B., Haas, P.H., Jochum, S., Margies, L., Müller, R.: Enhancing Assembly Instruction Generation for Cognitive Assistance Systems with Large Language Models. Procedia CIRP **134**, 7–12 (2025). https://doi.org/10.1016/j.procir.2025.03.010
9. Mattsson, S., Tarrar, M., Fast-Berglund, Å.: Perceived production complexity – understanding more than parts of a system **54**(20), 6008–6016. https://doi.org/10.1080/00207543.2016.1154210, https://www.tandfonline.com/doi/full/10.1080/00207543.2016.1154210
10. Pokorni, B., Popescu, D., Constantinescu, C.: Design of Cognitive Assistance Systems in Manual Assembly Based on Quality Function Deployment. Applied Sciences **12**(8), 3887 (Apr 2022). https://doi.org/10.3390/app12083887
11. Romero, D., Bernus, P., Noran, O., Stahre, J., Fast-Berglund, Å.: The Operator 4.0: Human Cyber-Physical Systems & Adaptive Automation Towards Human-Automation Symbiosis Work Systems. In: Nääs, I., Vendrametto, O., Mendes Reis, J., Gonçalves, R.F., Silva, M.T., Von Cieminski, G., Kiritsis, D. (eds.) Advances in Production Management Systems. Initiatives for a Sustainable World, vol. 488, pp. 677–686. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-51133-7_80
12. Stanton, N.A.: Hierarchical task analysis: Developments, applications, and extensions. Applied Ergonomics **37**(1), 55–79 (Jan 2006). https://doi.org/10.1016/j.apergo.2005.06.003
13. Sweller, J.: Element Interactivity and Intrinsic, Extraneous, and Germane Cognitive Load. Educational Psychology Review **22**(2), 123–138 (Jun 2010). https://doi.org/10.1007/s10648-010-9128-5
14. Thorvald, P., Fast Berglund, Å., Romero, D.: The Cognitive Operator 4.0. In: Shafik, M., Case, K. (eds.) Advances in Transdisciplinary Engineering. IOS Press (Aug 2021). https://doi.org/10.3233/ATDE210003
15. VDI 2860: Assembly and handling; handling functions, handling equipment; terms, definitions, symbols (May 1990), withdrawn in 2018; utilized here for its foundational taxonomic classification.