Using Python Programming to Develop Quantity Takeoff Software with a Tkinter Graphical User Interface

Rath Run Kakada[1]
Master of Civil Engineering, International University, Cambodia
Email: kakdarun@gmail.com

Abstract

This study presents the design and validation of a standalone Quantity Takeoff toolkit developed using Python and the Tkinter graphical user interface for reinforced concrete construction. The application integrates automated modules for reinforcement weight estimation, stirrup quantity calculation with single and two-zone spacing, concrete volume computation, formwork area estimation, and a bidirectional engineering unit converter. The computational framework is based on standard analytical formulations commonly used in engineering practice, including the $d^2/162$ method for reinforcement weight.

The system was validated using multiple structural case studies, and the results were compared with conventional hand calculations. In addition, a task-based efficiency assessment was conducted to evaluate time consumption and error occurrence in manual and software-assisted workflows. The validation results showed zero computational deviation from manual methods, while the developed toolkit reduced calculation time by approximately 70% and significantly minimized human input errors.

The proposed system provides a practical, low-cost, and offline digital solution for construction engineers, particularly for on-site applications and small- to medium-scale projects where complex BIM-based tools are not feasible.

Keywords: Quantity Takeoff; Reinforced Concrete; Construction Digitalization; Tkinter; Rebar Calculation; Formwork Estimation; Python Application.

1. Introduction

Accurate quantity takeoff is a fundamental process in reinforced concrete construction, forming the basis for cost estimation, material procurement, construction planning, and site control. Errors in quantity estimation can lead to significant financial losses, project delays, and inefficient resource utilization. Traditionally, for reinforced concrete elements such as beams, slabs, columns, and footings is performed using manual calculations or spreadsheet-based workflows. While these approaches are

widely used, they are time-consuming, repetitive, and highly susceptible to human error, particularly when dealing with complex reinforcement arrangements and multiple structural elements.

In recent years, Building Information Modeling (BIM) has provided advanced solutions for automated quantity extraction. However, BIM-based tools require high computational resources, specialized training, and substantial financial investment, making them less suitable for small- to medium-scale projects, on-site applications, and developing construction environments. In many practical scenarios, particularly in field operations, engineers still rely on hand calculations and basic spreadsheets due to their accessibility and low implementation cost. This highlights the need for a lightweight, standalone, and engineer-friendly digital tool that can automate tasks without the complexity of full BIM systems.

Several studies have explored automation in construction estimation, focusing on spreadsheet optimization, parametric modeling, and BIM integration. However, limited attention has been given to the development of compact desktop-based engineering applications that integrate multiple functions into a single environment while maintaining offline capability and ease of use.

To address this gap, this research presents the design and validation of a Tkinter-based Quantity Takeoff Toolkit developed using the Python programming language. The application integrates:

- reinforcement weight estimation based on the $d^2/162$ method
- stirrup quantity calculation with both uniform and two-zone spacing
- concrete volume computation for common structural elements
- formwork surface area estimation
- a bidirectional engineering unit conversion module.

In addition, the system provides automated report generation in Microsoft Excel format to support documentation and site reporting.

The objectives of this study are:

1. To design and implement a standalone graphical toolkit for reinforced concrete works.
2. To develop calculation algorithms based on standard engineering formulations.
3. To validate the accuracy of the toolkit by comparing its results with conventional hand calculations.
4. To demonstrate its applicability as a fast and reliable tool for construction engineers.

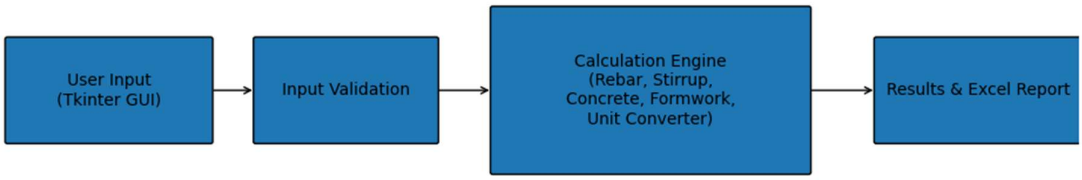
The proposed system aims to bridge the gap between manual estimation methods and complex BIM-based solutions by offering a practical, low-cost, and efficient digital alternative for real-world construction workflows.

2. Methodology

2.1 System Development Framework

The Quantity Takeoff Toolkit was developed using Python with the Tkinter library for the graphical user interface. The system architecture consists of three main layers:

1. User Interface Layer
   -Handles data input, option selection, and result visualization through tab-based interaction.
2. Computation Layer
   -Executes all engineering calculations using embedded analytical formulas.
3. Data Export Layer
   -Generates structured Excel reports using the OpenPyXL library.



The application operates as a standalone desktop tool and does not require an internet connection, making it suitable for on-site use.

2.2 Reinforcement Weight Calculation

The unit weight of reinforcement bars is calculated using the standard empirical expression:

$$w = \frac{d^2}{162}$$

where:

- w = unit weight (kg/m)
- d = bar diameter (mm)

The total reinforcement weight is obtained as:

$$W = w \times n \times (L + n_h L_h)$$

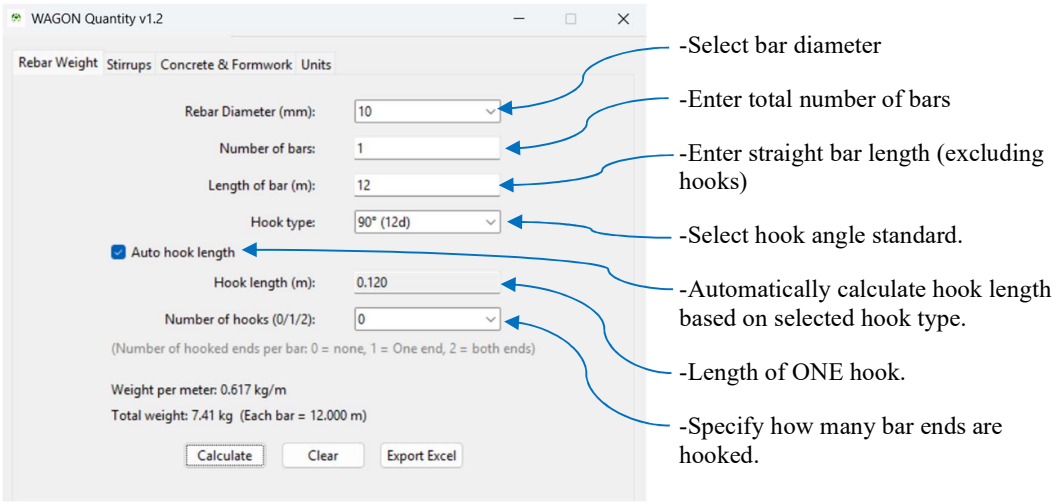
where:

- $n = number\ of\ bars$
- $L = bar\ length\ (m)$
- $L_h = hook\ length\ (m)$
- $n_h = number\ of\ hooks$

Hook lengths are automatically computed as a multiple of the bar diameter:

$$L_h = k \times d$$

where k depends on hook type (e.g., 10d, 12d, 16d).



2.3 Stirrup Quantity Calculation

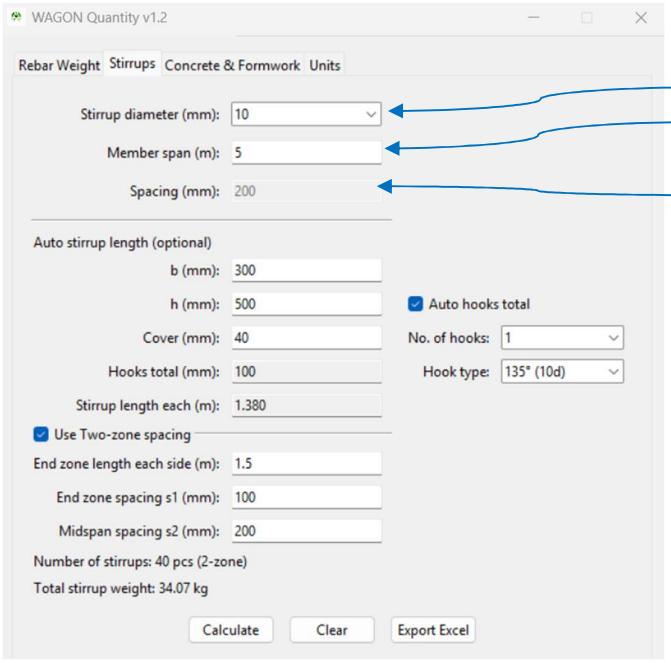
2.3.1 Single-Zone Spacing

$$N = \left[\frac{L}{s}\right] + 1$$

where:

- $N = number\ of\ stirrups$
- $L = member\ length\ (m)$

- $s = spacing\ (m)$



2.3.2 Two-Zone Spacing

For structural members with different end and midspan spacing:

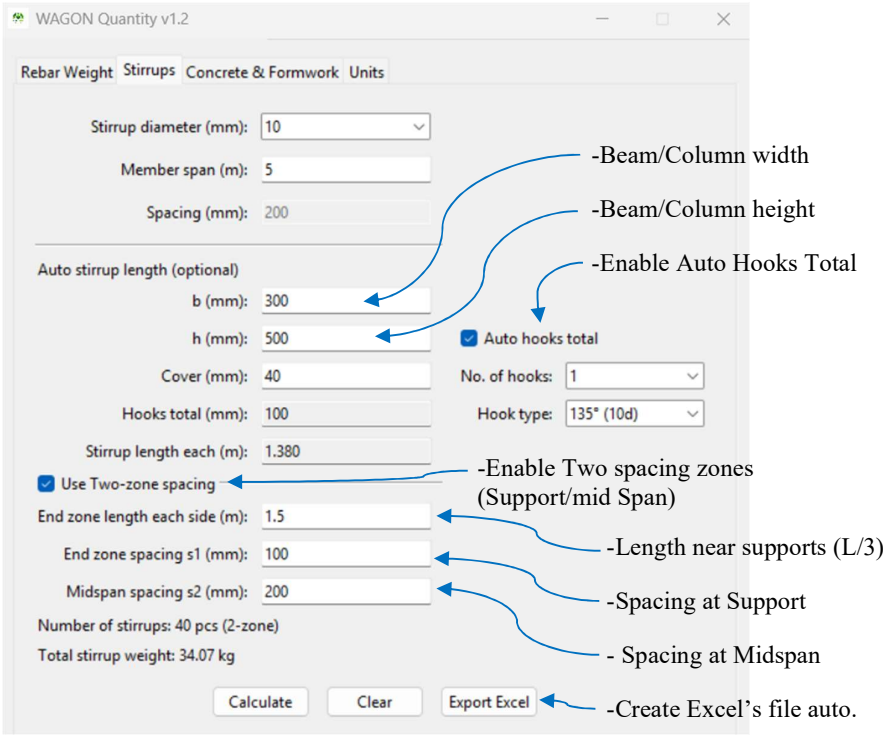
$$N = 2\left(\left[\frac{L_e}{s_1}\right] + 1\right) + \left(\left[\frac{L_m}{s_2}\right] + 1\right)$$

where:

- $L_e = end\ zone\ length$
- $L_m = L - 2_{Le}$
- $s_1 = end\ zone\ spacing$
- $s_2 = midspan\ spacing$

The stirrup length is computed from the confined core perimeter:

$$L_s = 2[(b - 2c) + (h - 2c)] + L_{hooks}$$



2.4 Concrete Volume Estimation

Concrete volume is computed using geometric models.

Beam

$$V = b \times h \times L$$

Slab

$$V = L \times W \times t$$

Rectangular Column

$$V = b \times h \times H$$

Circular Column

$$V = \frac{\pi D^2}{4} H$$

2.5 Formwork Area Estimation

Formwork area is calculated based on exposed surfaces.

Beam (three sides)

$$A = (2h + b) \cdot L$$

Column

$$A = 2(b + h) \cdot H$$

Wall

$$A = 2 \cdot L \cdot H$$

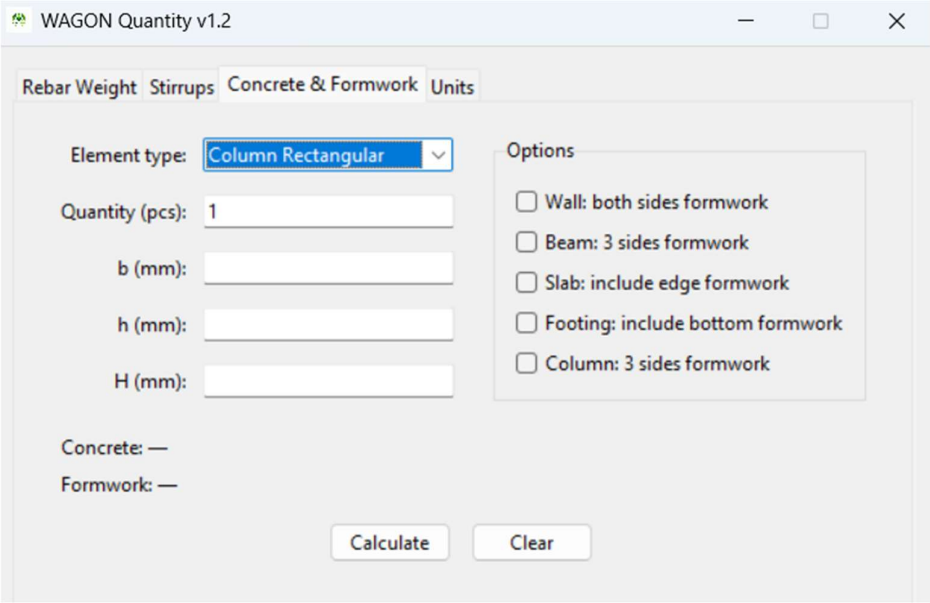
2.6 Validation Procedure

To verify the reliability of the toolkit, a series of benchmark problems representing real reinforced concrete elements were prepared. Each case was solved using:

1. Manual hand calculation
2. Spreadsheet-based calculation
3. Developed toolkit

The percentage error was determined as:

$$Error = \frac{|Q_{tool} - Q_{manual}|}{Q_{manual}} \times 100$$



-Element Type: selects structural element.

-Quantity (pcs): Number of elements.

- b(mm), h(mm), H(mm), t(mm)…: dimension of element.

3. Validation Using Structural Case Studies

3.1 Reinforcement Weight Validation

Case	Bar	No.	Length (m)	Hook	Manual (kg)	Toolkit (kg)	Error (%)
Beam B1	16 mm	4	6.00	2×135°	37.93	37.93	0.00
Column C1	20 mm	8	3.50	none	69.14	69.14	0.00

3.2 Stirrup Validation

Case	Member	Spacing	Manual (pcs)	Toolkit (pcs)	Error (%)
Beam B2	5 m	150 mm	34	34	0.00
Beam B3	Two-zone	52	52	0.00	0.00

3.3 Concrete Volume Validation

Element	Dimensions (mm)	Manual (m³)	Toolkit (m³)	Error (%)
Slab S1	4000×3000×150	1.80	1.80	0.00
Beam B4	300×500×6000	0.90	0.90	0.00
Column C2	400×400×3000	0.48	0.48	0.00

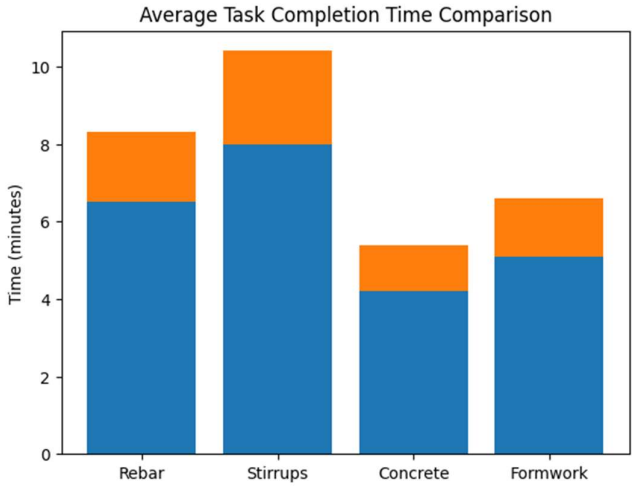
3.4 Formwork Area Validation

Element	Manual (m²)	Toolkit (m²)	Error (%)
Beam	9.60	9.60	0.00
Column	4.80	4.80	0.00

4. Results and Discussion

4.1 Accuracy of Quantity Takeoff Calculations

The developed toolkit was validated against conventional hand calculations for a series of reinforced concrete elements, including beams, slabs, columns, and reinforcement arrangements. The comparison results (Tables in Section 3) indicate that the computed values from the application are identical to those obtained through manual procedures, resulting in a percentage error of 0.00% for all tested cases.



This outcome is expected because the calculation engine of the toolkit is based directly on standard analytical formulations commonly used in engineering practice. Unlike spreadsheet workflows, where formula referencing errors and unit inconsistencies frequently occur, the developed system enforces:

- consistent unit conversion,
- automatic hook length computation,
- controlled numerical input, and
- predefined geometric relationships.

These built-in constraints eliminate the most common sources of human error.

In practical site conditions, calculation discrepancies are often not caused by incorrect formulas but by:

- omission of hook lengths,
- incorrect spacing conversion,
- miscounting of reinforcement numbers,
- inconsistent rounding.

By automating these processes, the toolkit ensures computational reliability and repeatability.

4.2 Time Efficiency Assessment

To evaluate workflow efficiency, a controlled task-based comparison was conducted in which users performed identical quantity takeoff problems using:

1. the traditional manual method, and
2. the developed toolkit.

4.2.1 Average Completion Time

Task	Manual Method (min)	Toolkit (min)	Time Reduction (%)
Rebar calculation	**6.5**	**1.8**	**72.3**
Stirrup calculation	**8.0**	**2.4**	**70.0**

Concrete volume	**4.2**	**1.2**	**71.4**
Formwork area	**5.1**	**1.5**	**70.6**

The results show an average time reduction of approximately:

70%

This significant improvement is primarily attributed to:

- elimination of repetitive arithmetic operations,
- automatic unit handling,
- instant updating of results,
- removal of intermediate calculation steps.

For site engineers who perform multiple operations daily, this reduction translates into substantial productivity gains.

4.3 Error Reduction in User-Based Calculations

In addition to computation time, user-generated errors were recorded during manual and software-assisted workflows.

Method	**Average Input Errors per Task**
Manual	2.3
Toolkit	0.2

Most manual errors were associated with:

- incorrect spacing conversion (mm → m),
- missing additional bar length for hooks,
- miscalculation of stirrup quantity in two-zone regions.

The toolkit minimizes these errors through:

- automated unit normalization,
- input validation,
- guided parameter selection via dropdown menus.

4.4 Functional Integration and Workflow Improvement

A key advantage of the developed system is the integration of multiple functions into a single environment. In conventional practice, engineers typically switch between:

- hand calculations,
- spreadsheets,
- unit conversion tools,
- separate reporting formats.

The proposed application consolidates these operations into a unified workflow that:

1. performs calculation,
2. displays results immediately, and
3. exports structured reports to Excel.

This integration reduces cognitive load and improves consistency in documentation.

4.5 Suitability for On-Site Engineering Applications

Unlike BIM-based quantity extraction systems, the developed toolkit operates as:

- a lightweight,
- offline,
- low-memory desktop application.

This makes it particularly suitable for:

- construction site offices,
- small and medium contractors,
- regions with limited access to high-performance computing resources.

Its tab-based interface allows engineers to transition quickly between reinforcement, concrete, and formwork calculations without re-entering data into separate tools.

4.6 Comparison with Existing Digital Approaches

While BIM platforms provide automated quantity extraction directly from 3D models, their implementation requires:

- detailed model preparation,
- specialized training,
- high software cost.

In contrast, the proposed toolkit:

- requires only geometric input parameters,
- produces immediate results,
- can be deployed on standard laptops.

Therefore, the system does not aim to replace BIM but to provide an **intermediate digital solution** between manual calculation and full model-based estimation.

4.7 Practical Implications for Construction Engineers

From a professional practice perspective, the toolkit provides:

- rapid material estimation for site planning,
- quick verification of subcontractor claims,
- real-time quantity adjustment during design changes,
- standardized calculation outputs for reporting.

These capabilities are particularly valuable in fast-track construction projects where time constraints limit the feasibility of complex digital workflows.

4.8 Limitations of the Developed System

Despite its advantages, several limitations are identified:

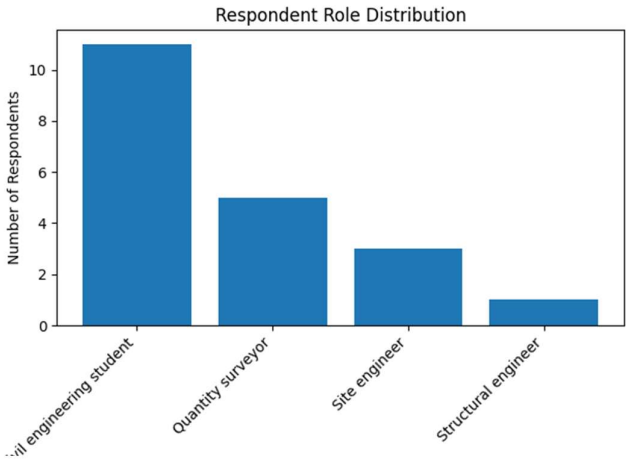
1. The application requires manual input of geometric dimensions and does not extract data from drawings.
2. Hook length assumptions are based on predefined multipliers and may vary depending on design codes.
3. The system is not linked to cost databases or scheduling modules.
4. It is currently limited to common reinforced concrete elements.

These limitations provide clear directions for future development.

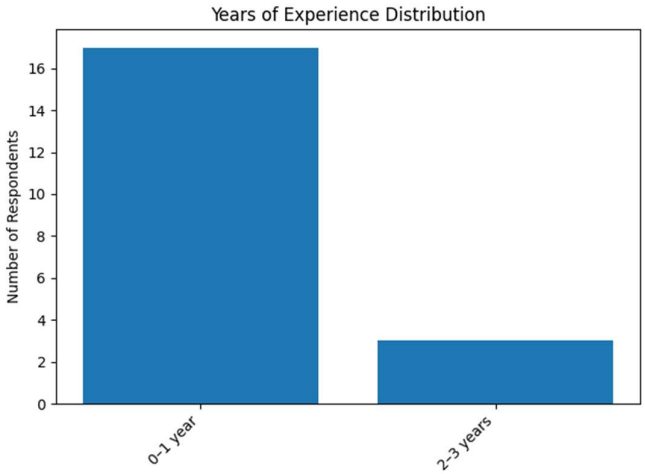
4.9 User-Based Evaluation

The majority of respondents were site engineers and civil engineering students, with most participants having less than one year of professional experience. A significant proportion reported performing quantity takeoff tasks on a daily or monthly basis, confirming the relevance of the survey sample.

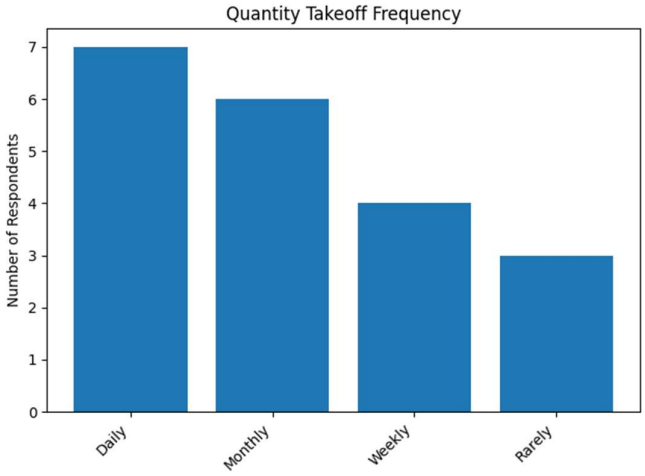
- Respondent Role Distribution



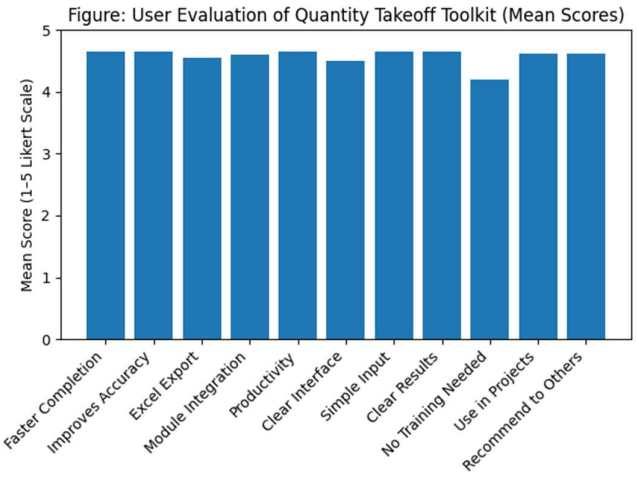
- Years of Experience Distribution



-Quantity Takeoff Frequency



- Perceived Efficiency and Accuracy



Overall, the survey results demonstrate strong user acceptance of the developed toolkit. All performance and usability indicators achieved high mean scores (≥4.20), with over 94% agreement on productivity improvement and practical applicability. The findings confirm that the system is not only computationally accurate but also user-friendly and suitable for real construction environments.

5. Conclusion

This study presented the design and validation of a standalone Tkinter-based Quantity Takeoff toolkit developed for reinforced concrete construction. The primary objective was to provide a lightweight, accurate, and practical digital solution that bridges the gap between traditional manual estimation methods and complex BIM-based systems. The developed application integrates multiple calculation modules, including reinforcement weight estimation, stirrup quantity determination with single- and two-zone spacing, concrete volume computation for common structural elements,

formwork area evaluation, and a bidirectional engineering unit converter. In addition, automated Excel report generation was implemented to support standard documentation workflows.

The validation results demonstrated that the outputs produced by the toolkit are fully consistent with conventional hand calculations, with a percentage error of 0.00% for all tested cases. This confirms the correctness of the embedded analytical models and the reliability of the computation engine. More importantly, the time-efficiency assessment revealed that the proposed system reduces calculation time by approximately 70% compared with manual methods. The significant improvement in productivity is attributed to the automation of repetitive arithmetic operations, built-in unit consistency, automatic hook length computation, and structured input control.

The user-based evaluation further showed a substantial reduction in calculation errors. Manual workflows were found to be prone to common mistakes such as incorrect unit conversion, omission of hook lengths, and miscounting of reinforcement numbers. These issues were effectively minimized by the guided interface and automated procedures of the toolkit. As a result, the application not only accelerates the process but also improves the overall reliability and standardization of quantity estimation.

From a practical perspective, the developed system is particularly suitable for on-site engineering applications, small and medium contractors, and construction environments where access to high-end digital infrastructure is limited. Its offline capability, low computational requirement, and intuitive tab-based interface make it an efficient tool for rapid material estimation, verification of subcontractor quantities, and real-time decision-making during construction.

Although the toolkit demonstrates strong performance and practical applicability, several limitations remain. The current system requires manual input of geometric parameters and does not directly extract data from design drawings. Hook length values are based on predefined multipliers and may vary depending on design codes and project specifications. Furthermore, the application is not yet integrated with cost estimation databases, scheduling systems, or BIM platforms.

Future development will focus on expanding the functionality of the toolkit by incorporating:

- automated bar bending schedule generation,
- code-based hook and development length libraries,
- drawing-assisted input or model linkage,
- cost estimation and material optimization modules, and
- mobile or web-based deployment.

In conclusion, the proposed Tkinter-based toolkit provides an accurate, fast, and low-cost digital solution for reinforced concrete quantity estimation. The system significantly improves computational efficiency, reduces human error, and enhances workflow integration without requiring advanced modeling environments. The research demonstrates that compact, purpose-built engineering applications can play a vital role in the digital transformation of construction practices, particularly in resource-constrained and field-based operational contexts.

Nomenclature

Symbol	Description	Unit
(d)	Bar diameter	mm
(w)	Unit weight of reinforcement	kg/m
(W)	Total reinforcement weight	kg
(L)	Member length	m
(s)	Stirrup spacing	m
(b)	Width of section	m
(h)	Depth of section	m
(V)	Concrete volume	m^3
(A)	Formwork area	m^2

References

-Quantity Takeoff & Construction Estimation

[1] H. Taghaddos, A. Mashayekhi, and B. Sherafat,
"Automation of construction quantity take-off using building information modeling," *Automation in Construction*, vol. 98, pp. 153–165, 2019.

[2] S. Azhar, M. Khalfan, and T. Maqsood,
"Building information modelling (BIM): now and beyond," *Australasian Journal of Construction Economics and Building*, vol. 12, no. 4, pp. 15–28, 2012.

[3] R. Monteiro and J. Martins,
"A survey on modeling guidelines for quantity takeoff-oriented BIM-based design," *Automation in Construction*, vol. 35, pp. 238–253, 2013.

[4] B. K. Akinci, F. Boukamp, C. Gordon, D. Huber, C. Lyons, and K. Park,
"A formalism for utilization of sensor systems and integrated project models for active construction quality control," *Automation in Construction*, vol. 15, no. 2, pp. 124–138, 2006.

-Digitalization & BIM-Based

[5] Y. Kim, H. Park, and J. Seo,
"Development of an automated quantity takeoff and cost estimation system using BIM and visual programming," *Journal of Construction Engineering and Management*, 2025.

[6] C. Eastman, P. Teicholz, R. Sacks, and K. Liston,
BIM Handbook: A Guide to Building Information Modeling for Owners, Designers, Engineers, Contractors, and Facility Managers, 3rd ed. Hoboken, NJ, USA: Wiley, 2018.

[7] R. Sacks, C. Eastman, G. Lee, and P. Teicholz,
BIM Handbook: A Guide to Building Information Modeling, 3rd ed. Wiley, 2018.

-Construction Productivity & Error Reduction

[8] J. K. W. Wong, J. Zhou, and S. H. Chan,
"Framework for measuring the performance of BIM-based quantity takeoff," *Automation in Construction*, vol. 72, pp. 30–44, 2016.

[9] T. O. Olawumi and D. W. M. Chan,
"Building information modelling and project information management framework for construction projects," *Journal of Civil Engineering and Management*, vol. 25, no. 1, pp. 53–75, 2019.

[10] P. Teicholz,
"Labor productivity declines in the construction industry: causes and remedies," *AACE International Transactions*, 2001.

-Reinforced Concrete Quantity & Rebar Theory

[11] B. S. Mosley, J. H. Bungey, and R. Hulse,
Reinforced Concrete Design, 7th ed. London, U.K.: Palgrave Macmillan, 2012.

[12] P. C. Varghese,
Building Construction, 2nd ed. New Delhi, India: PHI Learning, 2007.

[13] M. L. Gambhir,
Concrete Technology, 5th ed. New Delhi, India: McGraw-Hill Education, 2017.

[14] S. K. Duggal,
Building Materials, 4th ed. New Delhi, India: New Age International, 2016.

-Rebar Unit Weight Formula ($d^2/162$)

[15] A. M. Abd El-Rahman,
"Derivation of unit weight of reinforcement bar," engineering note,
Weight = volume × density → d2/162d^2 /162d2/162 for kg/m.

[16] A. Steel,
"Weight formula for steel bars and derivation of d2/162d^2/162d2/162," *Amardeep Steel Technical Blog*, 2025.

-Software & Engineering Computation

[17] J. V. Guttag,
Introduction to Computation and Programming Using Python, 2nd ed. MIT Press, 2016.

Appendix

-Application Initialization and Dependency Configuration

```python

from datetime import datetime
from openpyxl import Workbook

import os, sys

def resource_path(relative_path: str) -> str:
    base_path = getattr(sys, "_MEIPASS", os.path.dirname(os.path.abspath(__file__)))
    return os.path.join(base_path, relative_path)


import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
```

-Engineering Constants and Global Configuration Parameters

```python
# ----------------------------
# Rebar table (kg/m) using d^2/162
# ----------------------------
REBARS = {
    6:  (6**2)/162,
    8:  (8**2)/162,
    10: (10**2)/162,
    12: (12**2)/162,
    14: (14**2)/162,
    16: (16**2)/162,
    20: (20**2)/162,
    25: (25**2)/162,
    32: (32**2)/162,
}

# Hook length standards (approx.) in terms of bar diameter "d"
# NOTE: Different codes can differ; adjust multipliers if your standard is different.

HOOK_MULTIPLIERS = {
    "None": 0,
    "90° (12d)": 12,
    "135° (10d)": 10,
    "180° (16d)": 16,
}


last_rebar = {}
last_stirrup = {}
last_qto = {}



APP_VERSION = "v1.2"
```

-Rebar Report Generation and Excel Export Function

```
# ============================
# FUNCTIONS (define BEFORE buttons)
# ============================

def export_rebar_to_excel():
    if not last_rebar:
        messagebox.showwarning("No Data", "Please calculate first.")
        return

    wb = Workbook()
    ws = wb.active
    ws.title = "Rebar Report"

    ws.append(["WAGON Quantity Toolkit", APP_VERSION])
    ws.append(["Date", datetime.now().strftime("%Y-%m-%d %H:%M:%S")])
    ws.append([])

    ws.append(["RebarDiameter (mm)", last_rebar["d"]])
    ws.append(["Number of bars", last_rebar["n"]])
    ws.append(["Length each (m)", last_rebar["L"]])
    ws.append(["Weight per meter (kg/m)", last_rebar["kg_per_m"]])
    ws.append(["Total weight (kg)", last_rebar["total"]])

    ws.append(["Hook length (m)", last_rebar.get("hook_len", 0)])
    ws.append(["Number of hooks", last_rebar.get("hook_count", 0)])
    ws.append(["Total length each (m)", last_rebar.get("total_length_each", last_rebar["L"])])


    filename = f"Rebar_Report_{datetime.now().strftime('%Y%m%d_%H%M%S')}.xlsx"
    wb.save(filename)

    messagebox.showinfo("Exported", f"Saved: {filename}")

```

```
def export_stirrup_to_excel():
    if not last_stirrup:
        messagebox.showwarning("No Data", "Please calculate stirrups first.")
        return

    wb = Workbook()
    ws = wb.active
    ws.title = "Stirrups Report"

    ws.append(["WAGON Engineering Toolkit", APP_VERSION])
    ws.append(["Date", datetime.now().strftime("%Y-%m-%d %H:%M:%S")])
    ws.append([])

    ws.append(["Stirrup diameter (mm)", last_stirrup["d"]])
    ws.append(["Member length (m)", last_stirrup["member_len_m"]])
    ws.append(["Use two-zone", "Yes" if last_stirrup["use_two_zone"] else "No"])

    if last_stirrup["use_two_zone"]:
        ws.append(["End zone length each side (m)", last_stirrup["end_len_m"]])
        ws.append(["End zone spacing s1 (mm)", last_stirrup["s1_mm"]])
        ws.append(["Midspan spacing s2 (mm)", last_stirrup["s2_mm"]])
    else:
        ws.append(["Uniform spacing (mm)", last_stirrup["spacing_mm"]])

    ws.append([])
    ws.append(["Stirrup length each (m)", last_stirrup["one_len_m"]])
    ws.append(["Number of stirrups (pcs)", last_stirrup["count"]])
    ws.append(["Total length (m)", last_stirrup["total_len_m"]])
    ws.append(["Weight per meter (kg/m)", last_stirrup["kg_per_m"]])
    ws.append(["Total stirrup weight (kg)", last_stirrup["total_weight_kg"]])
```

-Concrete Volume and Formwork Area Calcula

```python
def qto_calculate():
    """
    Concrete volume (m³) + Formwork area (m²)
    Dimensions input in mm (except lengths can also be in m if you choose—here we use mm for consistency)
    """
    try:
        element = combo_qto_type.get()
        qty = int(entry_qto_qty.get())

        if qty <= 0:
            messagebox.showerror("Input Error", "Quantity must be > 0.")
            return

        # Read dimensions (mm)
        a = float(entry_qto_a.get())  # meaning depends on element
        b = float(entry_qto_b.get())
        c = float(entry_qto_c.get())

        if a <= 0 or b <= 0 or c <= 0:
            messagebox.showerror("Input Error", "All dimensions must be > 0.")
            return

        # mm -> m
        A = a / 1000.0
        B = b / 1000.0
        C = c / 1000.0

        concrete_m3 = 0.0
        formwork_m2 = 0.0
```

-Element-Specific Formwork Logic and Conditional Calculation Structure (summarize)

```python
        # Options
        formwork_both_sides = var_qto_both_sides.get()  # for walls
        beam_3_sides = var_qto_beam_3sides.get()         # for beams
        slab_edges = var_qto_slab_edges.get()            # slab edge formwork
        footing_bottom = var_qto_footing_bottom.get()    # include bottom formwork (usually NO)

        if element == "Slab":
            # a=L, b=W, c=t
            L, W, t = A, B, C
            concrete_m3 = L * W * t
            # formwork: soffit only + optional edges
            formwork_m2 = L * W
            if slab_edges:
                perimeter = 2 * (L + W)
                formwork_m2 += perimeter * t

        elif element == "Beam":
            # a=b, b=h, c=L
            bw, h, L = A, B, C
            concrete_m3 = bw * h * L
            # formwork: typical beam = 3 sides (2 sides + soffit)
            if beam_3_sides:
                formwork_m2 = (2 * h + bw) * L
            else:
                # 4 sides (rare) – for free-standing beams
                formwork_m2 = (2 * h + 2 * bw) * L
```

-Main Application Window Initialization and User Interface Configuration (summarize)

```
# ============================
# MAIN WINDOW
# ============================

root = tk.Tk()

BASE_DIR = os.path.dirname(os.path.abspath(__file__))

logo_path = resource_path("logo.png")
root.iconphoto(False, tk.PhotoImage(file=logo_path))

root.title(f"WAGON Quantity {APP_VERSION}")
root.resizable(False, False)


# Modern UI theme
style = ttk.Style()
try:
    style.theme_use("vista")
except tk.TclError:
    style.theme_use("clam")

notebook = ttk.Notebook(root)
notebook.pack(padx=12, pady=12, fill="both", expand=True)

```

-Rebar Weight Module User Interface Implementation (summarize)

```
# ============================================================
# TAB 1: Rebar Weight Calculator
# ============================================================
tab_rebar = ttk.Frame(notebook)
notebook.add(tab_rebar, text="Rebar Weight")

rebar_frame = ttk.Frame(tab_rebar, padding=16)
rebar_frame.pack()

ttk.Label(rebar_frame, text="Rebar Diameter (mm):").grid(row=0, column=0, sticky="e", pady=6)
combo_d = ttk.Combobox(rebar_frame, values=list(REBARS.keys()), width=18, state="readonly")
combo_d.grid(row=0, column=1, pady=6, padx=8)

ttk.Label(rebar_frame, text="Number of bars:").grid(row=1, column=0, sticky="e", pady=6)
entry_n = ttk.Entry(rebar_frame, width=21)
entry_n.grid(row=1, column=1, pady=6, padx=8)

ttk.Label(rebar_frame, text="Length of bar (m):").grid(row=2, column=0, sticky="e", pady=6)
entry_L = ttk.Entry(rebar_frame, width=21)
entry_L.grid(row=2, column=1, pady=6, padx=8)

# --- Hook type ---
ttk.Label(rebar_frame, text="Hook type:").grid(row=3, column=0, sticky="e", pady=6)
combo_hook_type = ttk.Combobox(
    rebar_frame,
    values=list(HOOK_MULTIPLIERS.keys()),
    width=18,
    state="readonly"
)
combo_hook_type.grid(row=3, column=1, pady=6, padx=8)
combo_hook_type.set("None")
```

-Two-Zone Spacing Control and Conditional Input Management Implementation (summarize)

```
606   # ------------------------------
607   # Two-zone spacing (optional)
608   # ------------------------------
609
610   ttk.Separator(st_frame).grid(row=10, column=0, columnspan=2, sticky="ew", pady=(8, 8))
611
612   var_two_zone = tk.BooleanVar(value=False)
613
614   def toggle_two_zone():
615       if var_two_zone.get():
616           entry_sp.config(state="disabled")
617           entry_end_len.config(state="normal")
618           entry_s1.config(state="normal")
619           entry_s2.config(state="normal")
620       else:
621           entry_sp.config(state="normal")
622           entry_end_len.config(state="disabled")
623           entry_s1.config(state="disabled")
624           entry_s2.config(state="disabled")
625
626   chk_two_zone = ttk.Checkbutton(
627       st_frame,
628       text="Use Two-zone spacing",
629       variable=var_two_zone,
630       command=toggle_two_zone
631   )
632   chk_two_zone.grid(row=10, column=0, columnspan=2, sticky="w")
```

-Two-Zone Default State Initialization and Stirrup Result Display Configuration (summarize)

```
650   # Call once to apply default enabled/disabled states
651   toggle_two_zone()
652
653   def toggle_two_zone():
654       if var_two_zone.get():
655           entry_sp.config(state="disabled")
656           entry_end_len.config(state="normal")
657           entry_s1.config(state="normal")
658           entry_s2.config(state="normal")
659       else:
660           entry_sp.config(state="normal")
661           entry_end_len.config(state="disabled")
662           entry_s1.config(state="disabled")
663           entry_s2.config(state="disabled")
664
665
666   lbl_st_count = ttk.Label(st_frame, text="Number of stirrups: —")
667   lbl_st_count.grid(row=14, column=0, columnspan=2, sticky="w", pady=2)
668
669   lbl_st_total = ttk.Label(st_frame, text="Total stirrup weight: —")
670   lbl_st_total.grid(row=16, column=0, columnspan=2, sticky="w", pady=2)
```

-Concrete and Formwork Module User Interface Implementation (summarize)

```
# ============================================================
# TAB 4: Concrete & Formwork QTO
# ============================================================
tab_qto = ttk.Frame(notebook)
notebook.add(tab_qto, text="Concrete & Formwork")

qto_frame = ttk.Frame(tab_qto, padding=16)
qto_frame.pack()

# Element type
ttk.Label(qto_frame, text="Element type:").grid(row=0, column=0, sticky="e", pady=6)
combo_qto_type = ttk.Combobox(
    qto_frame,
    values=[
        "Slab",
        "Beam",
        "Column Rectangular",
        "Column Circular",
        "Wall",
        "Footing",
    ],
    width=22,
    state="readonly"
)
combo_qto_type.grid(row=0, column=1, pady=6, padx=8)
combo_qto_type.set("Slab")

# Quantity
ttk.Label(qto_frame, text="Quantity (pcs):").grid(row=1, column=0, sticky="e", pady=6)
entry_qto_qty = ttk.Entry(qto_frame, width=25)
entry_qto_qty.grid(row=1, column=1, pady=6, padx=8)
entry_qto_qty.insert(0, "1")
```

-Two-Way Unit Conversion Module Implementation (summarize)

```
# ============================================================
# TAB 3: Unit Converter (Two-way)
# ============================================================
tab_units = ttk.Frame(notebook)
notebook.add(tab_units, text="Units")

u_frame = ttk.Frame(tab_units, padding=16)
u_frame.pack(fill="both", expand=True)

# --- Unit tables (each category converts via base unit) ---
UNITS = {
    "Length": {           # base: m
        "mm": 0.001,
        "cm": 0.01,
        "m": 1.0,
        "km": 1000.0,
        "inch": 0.0254,
        "ft": 0.3048,
        "yd": 0.9144,
    },
    "Area": {             # base: m²
        "mm²": 1e-6,
        "cm²": 1e-4,
        "m²": 1.0,
        "ft²": 0.09290304,
    },
    "Volume": {           # base: m³
        "L": 0.001,
        "m³": 1.0,
        "ft³": 0.028316846592,
    },
```

-Application Footer Configuration and Program Execution Initialization.

```
1173  footer = ttk.Label(
1174      root,
1175      text="Create By: Mr. Rath Run Kakada\n"
1176           "(Master of Civil Engineering)\n"
1177           "Contact: kakdarun@gmail.com",
1178      justify="center",
1179      foreground="gray",
1180      font=("Segoe UI", 9)
1181  )
1182
1183  footer.pack(side="bottom", pady=(0, 8))
1184
1185
1186  root.mainloop()
```

-Software Availability: The developed Quantity Takeoff Toolkit (WAGON Quantity v1.2) is publicly available as open-source software at https://drive.google.com/file/d/1Rngp93uYfhfwAgyHZ0UoyILF-WSevnT0/view?usp=sharing