LOC vs FP vs COCOMO : A Comparative Study

INTRODUCTION

Estimating software development effort is essential for planning and resource allocation. However, traditional models like LOC and Function Points struggle with the unique structure of Object-Oriented (OO) software, which features encapsulation, inheritance, and polymorphism.

OBJECTIVE

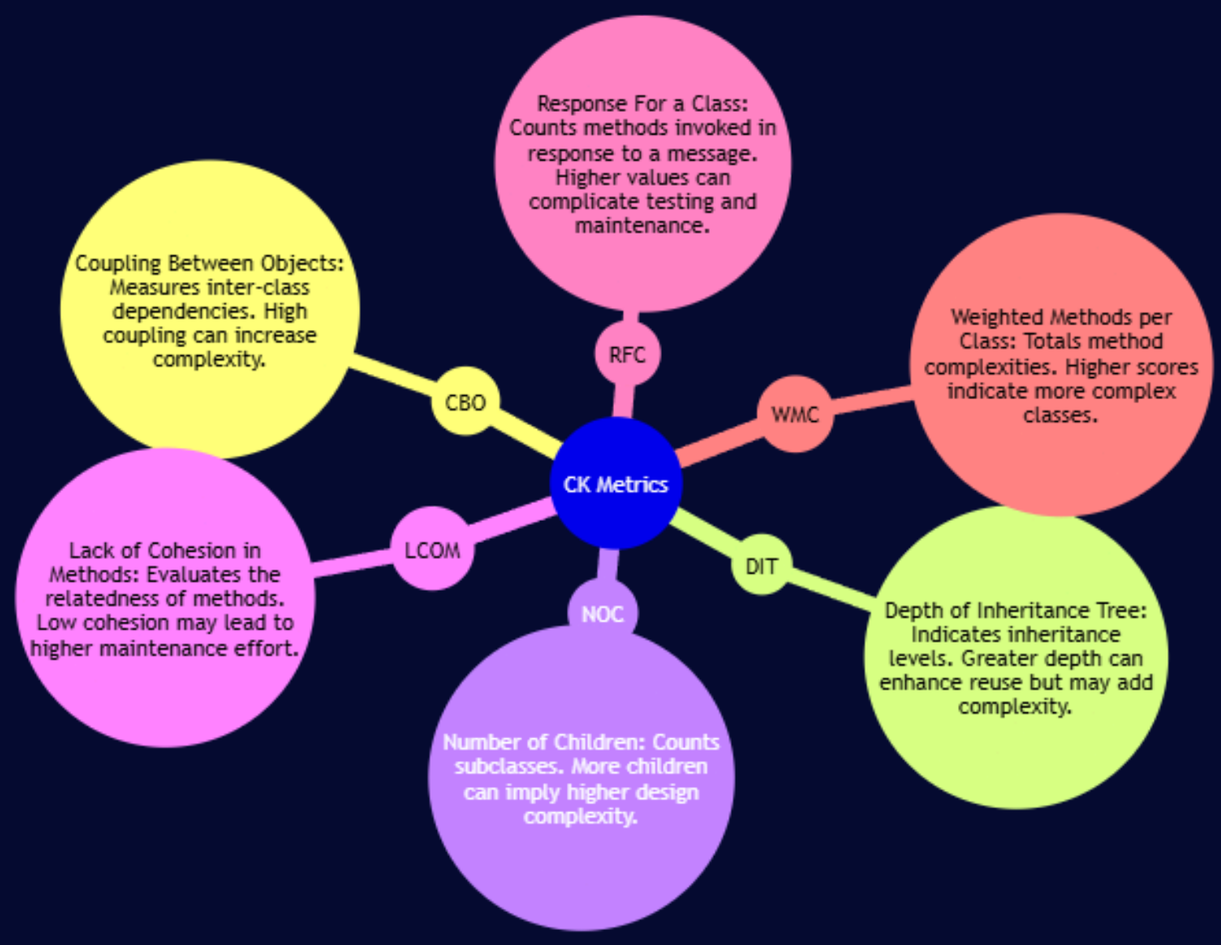
The paper acknowledges that traditional effort estimation methods (like LOC and Function Points) often fall short when applied to object-oriented (OO) software due to its unique features such as encapsulation, inheritance, and polymorphism. This necessitates tailored estimation techniques for OO projects.

COMPARISON

Model	Early Estimation	OO Suitability	Customise	Accuracy
LOC	No	No	Low	Low
FP	Yes	Partial	Medium	Medium
COCOMO II	Yes	Yes	High	High
Hybrid	Yes	Yes	Very High	Very High

OBJECT-ORIENTED METRICS AND THEIR ROLE

Object-Oriented design adds complexities not captured by traditional estimation models. CK Metrics address this by measuring OO-specific traits like coupling, inheritance, and cohesion. Metrics such as CBO, DIT, NOC, and LCOM quantify inter-class dependencies, hierarchy depth, subclassing, and method cohesion. These insights improve estimation accuracy by reflecting structural and maintenance-related factors in OO software.



LINES OF CODE (LOC)

LOC estimates software size by counting lines of code, making it simple but language-dependent and inconsistent. It doesn't account for complexity or productivity in OO systems and is less useful in early project phases.

FUNCTION POINT ANALYSIS (FPA)

FPA measures functionality from a user's perspective, independent of programming language and useful in early development. However, it lacks a direct mapping to object-oriented constructs and needs adaptation.

COCOMO (CONSTRUCTIVE COST MODEL):

COCOMO estimates cost based on project size using Basic, Intermediate, and Detailed models. Though helpful for traditional projects, it needs modification to suit object-oriented development.

COCOMO II:

COCOMO II supports modern practices like reuse and prototyping, making it more flexible for OO systems. It introduces refined cost drivers but still requires customization for accurate estimations.

CONCLUSION

Traditional software estimation models like LOC, FPA, and COCOMO, while foundational, fall short when applied to the complexities of object-oriented software. Object-oriented systems demand more nuanced approaches that account for features like inheritance, encapsulation, and polymorphism. Integrating OO-specific metrics such as those in the CK suite with traditional models—especially COCOMO II—significantly improves estimation accuracy. Additionally, leveraging early-stage design artifacts (e.g., UML) and advanced techniques like machine learning offers a more adaptive and reliable path for estimating effort in OO development.

TEAM

16010422277-Soham Mahendra Bagal
16010422242-Heramb Rajendra Patil
16010422225- Atul Chatti
16010422205- Parth Vyas
16010422097- Aarush Saboo
16010422146 - Sanika Tope