Abstrak

Design pattern muncul akibat adanya permasalahan yang sama yang sering muncul pada desain pembuatan perangkat lunak. Pada perkembangannya sudah banyak *design pattern* yang sudah ditemukan oleh pala programmer. Saat ini, *design pattern* dikelompokkan ke dalam tiga tujuan berbeda, yaitu *creational, structural,* dan *behavioral.*

State pattern merupakan salah satu *design pattern* yang tergolong ke dalam *behavioral pattern. State pattern* muncul akibat adanya kondisi-kondisi (*states*) yang muncul pada saat pembuatan perangkat lunak. *Pattern* ini mengijinkan *state transition logic* untuk disatukan dengan sebuah *state object* daripada berada dalam kondisional atau *switch statement.*

Pada tugas akhir kali ini, dibuat sebuah perangkat lunak yang mengimplementasikan *state pattern* untuk menyelesaikan sebuah kasus yang memiliki beberapa kondisi. Untuk mengevaluasi *state pattern*, dilakukan pengujian serta perhitungan terhadap *object-oriented metrics*. Sedangkan untuk mengetahui kelebihan dan kekurangan yang ada pada *state pattern*, akan dibandingkan hasil perhitungan *object-oriented metrics* perangkat lunak yang menerapkan *state pattern* dengan perangkat lunak yang tidak menerapkan *state pattern* untuk sebuah kasus yang serupa.

Berdasarkan hasil analisis dan pengujian, jika dilihat dari perhitungan *object-oriented metrics*, perangkat lunak dengan *state pattern* memiliki kompleksitas yang lebih tinggi dan membutuhkan usaha yang lebih besar ketika dilakukan *maintenance* dibandingkan perangkat lunak tanpa *state pattern*. Namun, jika dilihat dari penanggulangan *state* yang ada, perangkat lunak dengan *state pattern* memiliki cara penanggulangan *state* yang lebih optimal dibandingkan perangkat lunak tanpa *state pattern.*

Kata kunci: design pattern, state pattern, dan object-oriented metrics.