ABSTRAK

Banyak pengalaman yang menunjukkan bahwa rancangan perangkat lunak yang tidak berorientasi objek mengalami kesulitan dalam beradaptasi terhadap perubahan sistem yang ada. Perangkat lunak semacam ini ini sulit dikembangkan karena program sulit untuk diubah atau dikoreksi. Hal ini terjadi karena banyak pengembang perangkat lunak boleh jadi masih awam terhadap bagaimana suatu masalah dianalisa, dimodelkan dan dirancang dengan berorientasi objek. Ada beberapa metodologi pengembangan perangkat lunak yang dapat digunakan untuk memecahkan permasalahan yang ada sehingga mudah diimplementasikan ke dalam suatu bahasa pemrograman. Salah satu metodologi adalah OMT (*Object Modelling Technique*) yang diciptakan Rumbaugh.

Studi literatur ini bertujuan untuk mempelajari bagaimana konsep teknik pemodelan berorientasi objek pada pengembangan perangkat lunak dengan menggunakan metodologi OMT serta melakukan penerapan metodologi tersebut pada lima buah contoh permasalahan.

Empat langkah utama yang merupakan metode dalam OMT diawali dengan langkah **analisis**, dimana model abstraksi dibuat dari apa yang dilakukan oleh sistem. Pada langkah ini mulai dibuat beberapa diagram yang mendefinisikan struktur dasar dari sistem tanpa perlu terlalu awal memikirkan bagaimana mengimplementasikannya. Langkah kedua adalah **perancangan sistem**, bentuk keputusan (*architecture decision*) dibuat secara detail tentang sistem. Ketiga adalah **perancangan objek**, pada langkah ini implementasi didefinisikan secara detail dari setiap langkah sebelumnya. Digunakan diagram yang sama seperti pada langkah analisis, tetapi dibuat lebih detail lagi. Langkah yang terakhir adalah **implementasi**, dimana sistem diimplementasikan ke dalam suatu bahasa pemrograman berdasarkan langkah ketiga

Sedangkan tiga jenis model utama yang ersedia dalam OMT adalah **model objek**, yang menggambarkan gambaran statis struktur sistem; **Model dinamik**, yang menggambarkan hubungan dinamik antara sistem dan dunia luar. Model dinamik dapat terdiri dari banyak diagram. Yang pertama adalah diagram *state*, yang memberikan pada kita tentang gambaran dinamik dari apa yang terjadi pada sebuah class. Model dinamik mungkin juga memiliki diagram *event trace* dan diagram *event flow*, yang memberikan gambaran mengenai sistem dari sisi dinamik; Dan yang terakhir adalah **model fungsional**, yang menunjukkan perubahan nilai-nilai dalam sistem. Model ini menggunakan notasi DFD tradisional.

Dengan maksud membuktikan bahwa OMT mampu menyelesaikan beberapa jenis aplikasi yang berbeda, maka lima contoh permasalahan diambil untuk dimodelkan dengan menggunakan metodologi OMT, yaitu (1) sistem ATM (*embedded micro application*), (2) animasi komputer (*computer graphic*), (3) *object diagram compiler (compiler)*, (4) sistem perpustakaan (*database management system*), dan (5) sistem agen properti (*business application*).

Dengan menggunakan pemodelan OMT, seorang pemrogram akan dapat mengimplementasikan sistem dengan mudah, karena diketahui dengan jelas *framework* dari sistem. Dalam banyak kasus, *programmer* sering menghasilkan

aplikasi yang kurang baik, karena kurangnya pemahaman permasalahan yang dihadapi. Sedangkan dengan pemodelan OMT, permasalahan itu akan dapat digambarkan dengan lengkap. Selama pemodelan sistem, tidak diperhatikan bagaimana nantinya sistem diimplementasikan, karena hasil pemodelan dan perancangan dengan menggunakan metodologi OMT dapat diimplementasikan dengan menggunakan OOP (*Object Oriented Programming)* ataupun non-OOP.

Diasumsikan pembaca tugas akhir ini memahami konsep dasar paradigma berorientasi objek.