International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 24 Issue 7 – APRIL 2017 (SPECIAL ISSUE)

Component Based Adaptation by Configuration Management and Component Testing For Video Surveillance

R.SARADHA [#1] and Dr. X. MARY JESINTHA[*2]

[#] *Research Scholar, Bharathiar University,Assistant Professor,Saradha Gangadharan College,Pondicherry, India.*

[*] *Research Supervisor, Bharathiar University,Professor, Department of MCA,Vivekananda Institute of Engineering,Thiruchengode, India*

***Abstract*— The main objectives of this research paper are System adaptation based on the specification, measurement and optimization of quality attribute properties on feature models, Provide efficient video surveillance system specification and image acquisition, classification, segmentation, shadow removal. The Methods and Statistical analysis used in this research are Development of a method used to build prediction-enabled component technologies, Development of a validation method for component technologies, Implementation of component technologies according to the defined method for a particular quality attributes and Validation of the implemented component technologies using the defined method. The main Findings of the research are Classification of quality attributes, Demonstration of how to achieve predictability for another quality attributes an analysis of the suitability of predictability in industrial settings and Demonstration of a component model with capabilities for predicting consistency between components.**

***Index Terms*— Component based adoption, configuration management, and component testing**

I. INTRODUCTION

Programming frameworks are turning out to be progressively mind boggling and giving greater usefulness. To have the capacity to deliver such frameworks cost-viably, providers regularly utilize segment based advancements as opposed to building up every one of the parts of the framework starting with no outside help. The inspiration driving the utilization of segments was at first to diminish the cost of advancement, yet it later turned out to be more essential to lessen an opportunity to market, to meet quickly developing shopper requests. At present, the utilization of segments is all the more frequently roused by conceivable decreases being developed expenses. By utilizing parts it is conceivable to create greater usefulness with a similar venture of time and cash. At the point when segments are presented in a framework, new issues must be managed e.g. dynamic arrangements, variation blast and adaptability. Some of these issues are tended to with the train Component-Based Software Engineering (CBSE). CBSE gives strategies, models and rules for the engineers of segment based frameworks. Segment based improvement (CBD) means the advancement of frameworks making significant utilization of parts.

Feature modeling has been widely used in domain engineering for the development and configuration of software product lines. A feature model represents the set of possible products or configurations to apply in a given context. Recently, this formalism has been applied to the runtime configuration of systems with high variability and running in changing contexts. These systems must adapt by updating their component assembly configuration at runtime, while minimizing the impact of such changes on the quality of service. For this reason the selection of a good system configuration is seen as an optimization problem based on quality attribute criteria.

A feature model is arranged in a hierarchy that forms a tree where features are connected by:

- Tree constraints: relationships between a parent feature and its child features (or sub-features). Tree constraints include mandatory, optional, xor (alternative) and or relationships between parents and sub-features.
- Cross–tree constraints: typically inclusion or exclusion statements of the form “if feature F is selected, then features A and B must also be selected (or deselected)”.

The root feature of the tree represents the concept being described, generally the system itself, and the remaining nodes denote branches and sub-features that dis-aggregate the main concept into several elements and concerns.

One of the basic problems when developing component-based systems is that it is difficult to keep track of components and their interrelationships. This is particularly problematic when upgrading components. One way to maintain control over upgrades is to use component identification and dependency analysis. These are well known techniques for managing system configurations during development, but are rarely applied in managing run-time dependencies.

II. LITERATURE REVIEW

Sommerville 2010 Modern patterns in CBSE innovation empower principle points of interest of programming reuse. These points of interest incorporate upgrades in quality, exertion (cost) and time-to-market, and gauges.

VanVliet 2008 Software advancement is unavoidable in any product framework since changes in the public eye and innovation will require consequent changes to programming frameworks to stay up with the latest

In addition, proficiency in the product procedure is vital because of the perpetually expanding request on accessible improvement limit. The collaboration fundamental in programming building impacts e.g. the dissemination of work, correspondence, norms and methods.

Include models (Kang et al. 1990) are a basic yet capable formalism for speaking to shared characteristics, differing angles, and design guidelines of programming items, which have been for the most part utilized as a part of Software Product Lines (SPLs). In late works, include models have been connected for determining and executing progressively versatile frameworks. These frameworks can be conceptualized as a dynamic programming product offering (DSPL) (Hallsteinsen et al. 2008) in which inconstancy and arrangement principles are bound and checked at runtime. As in customary SPLs, include models are an advantageous formalism for speaking to a DSPL and empower robotized thinking about properties of its versatile arrangements. A versatile framework is a framework whose conduct can be changed amid its execution as indicated by the client's needs or setting changes. In the event that the framework can respond to changes in the working condition, the framework is called self-versatile (Oreizy et al. 1999). In (Moisan et al. 2011), include models were proposed for the portrayal and element adjustment of segment based frameworks, for example, a video reconnaissance (VS) preparing chain. The space of PC vision and video reconnaissance offers a testing ground as a result of the high inconstancy in both the observation undertakings and the video examination calculations. From a utilitarian viewpoint, the different VS assignments (e.g., tallying, interruption location, following, situation acknowledgment) have distinctive necessities, in particular perception conditions, objects of intrigue, and gadget arrangements, among others; which may fluctuate starting with one application then onto the next. From an execution point of view, choosing the (product) segments themselves, gathering them, and tuning their parameters to follow the setting may prompt to various setup variations. Additionally, the setting is not settled but rather advances powerfully and along these lines requires runtime adjustment of the part get together to continue performing with an alluring nature of service. (Sanchez et al. 2013), we displayed a heuristic inquiry calculation called CSA (Configuration Selection Algorithm)1 for taking care of the advancement issue coming about because of choosing a legitimate setup of a framework in view of highlight models. This calculation offers diverse systems for utilizing execution productivity and optimality, and permits us to characterize distinctive target capacities for looking at setups and enhancing different properties at the same time, while sticking to asset limitations and highlight show imperatives. Be that as it may, this calculation requires a foundation with capacities for: observing setting changes, enacting and amassing (at runtime) segments that actualize particular components, and social occasion reasonable measurements for framework properties, in order to survey different setup choices. (Sanchez et al. 2014). Contrasted with before work, we give extra data about our approach, its application, and executing stage.

This paper exhibits a strategy for investigating conditions between segments. The technique predicts the impact of a segment refresh by recognizing the parts in a framework and building a diagram depicting their conditions. Information of the conceivable impacts of a refresh is imperative, since it can be utilized to confine the extent of testing and be a reason for assessing the potential harm of the refresh. The reliance diagrams can likewise be utilized to encourage upkeep by distinguishing contrasts between designs, e.g., making it conceivable to perceive any deviations from a working reference setup.

III. OBJECTIVES

The goal of our model-based approach for managing quality attributes is to quantitatively evaluate and trade-off multiple quality attributes to achieve a better overall system configuration. We do not look for a single metric but rather for a quantification of individual attributes and for trade-offs among those metrics

We propose an approach for system adaptation based on the specification, measurement and optimization of quality attribute properties on feature models , Configuration management, configuration component testing . Furthermore, we describe its integration into a platform for supporting the self-adaptation of component-based systems. Feature models are annotated with quality attribute properties and metrics, and then an efficient algorithm is used to deal with the optimization problem.

IV. DESIGN AND DEVELOPMENT OF COMPONENT BASED ADAPTION

We present the overall approach and the component-based platform in which the CSA is embedded. Our approach provides a framework for the specification; measurement and optimization of quality attribute properties[2] expressed on top of feature models. We show how these properties can be specified by means of feature attributes and evaluated with *quality metrics* in the context of feature models. The global properties of the system are computed by means of aggregate functions over the features. Along this line, we discuss the selection process carried out by our optimization algorithm, highlighting some trade-off situations between quality attributes. A key aspect of model-based approaches for adaptive systems is the ability of the model to estimate a given system property, which is correlated with the actual property observed in the running system. For instance, if our approach computes a metrics for reconfiguration time as the sum of the individual times for each reconfiguration operation, e.g., add or remove a component from system assembly, we need to ensure that the aggregate metrics is a “good predictor” for the time that the system takes to

reconfigure itself.

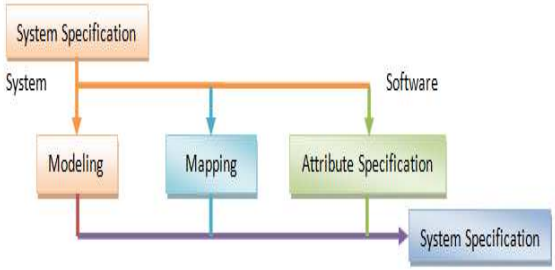

Figure 1: General Processing Stages

We also extend its evaluation with new experiments that assess the accuracy of the proposed metrics. To do so, we rely on a concrete implementation of our platform for managing the adaptation of a computer vision processing chain based on OpenCV libraries (Opencv project 2015). This processing chain includes components for image segmentation, motion, face detection, etc. In particular, we are focused on two properties –*reconfiguration time* and *frame processing time*– which are common in computer vision systems, and then compare predicted against measured property values. Our experiments reported an accuracy of 87.6 % and 90.6 % for these two properties, respectively. These preliminary results suggest that it is possible to predict quality attribute properties with simple aggregate functions defined on feature models.

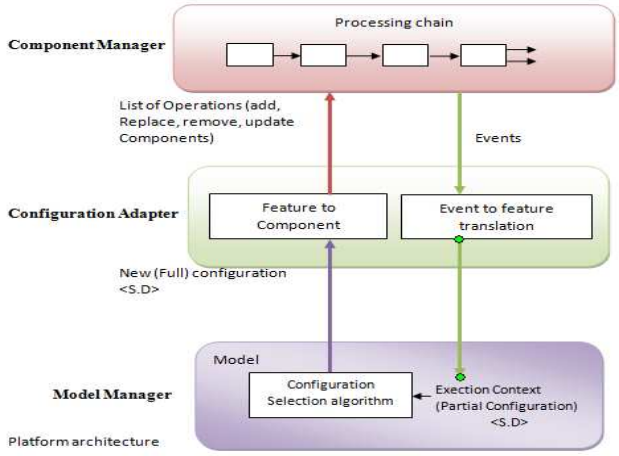

Figure 2: The three stage Model

We evaluated the optimization aspect of the approach by conducting some experiments in which we analyzed scalability, efficiency, and optimality of CSA using automatically generated scenarios. Evaluation is done with concrete measures and analyzes the accuracy of the additive and maximum metric functions for estimating two properties of interest in a video processing chain: *frame processing time* and *reconfiguration time*. Specifically, we performed four experiments with the additive and maximum metrics: two for frame processing time, and other two for reconfiguration time. The goal was to compare predicted against measured properties of the running system. To do so, we stated the following research directions

- The average accuracy of the additive metric for predicting frame processing time in sequential execution
- The average accuracy of the maximum metric for predicting frame processing time in parallel execution
- The average accuracy of the additive metrics for predicting reconfiguration time

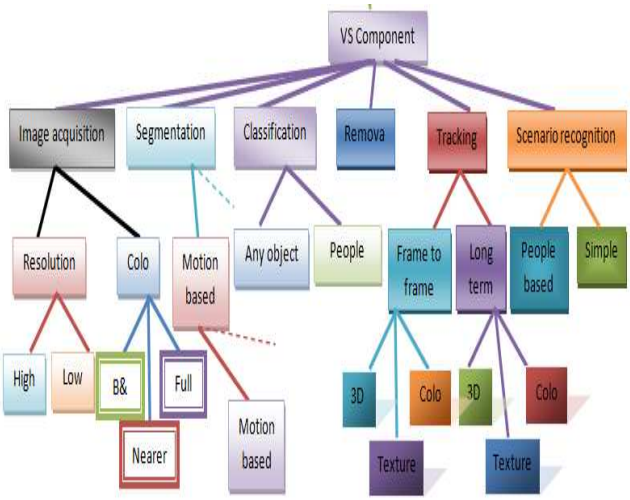

Figure3: Overall Process stages in proposed architecture

V. PERFORMANCE EVALUATION

Straight connection amongst deliberate and anticipated estimations of edge preparing time and reconfiguration time properties. Relapse lines (dabbed dark bends) marginally digress from the perfect pattern (strong red bends) that speaks to the ideal match of assessed and genuine property estimations. Other than direct connection, we broke down the Accuracy of the measurements. The precision is a proportion in the vicinity of 0 and 1 that is registered as 1 − Fault Rate, being the blame rate the normal relative contrast amongst assessed and real property estimations:

Metric is sufficient for evaluating outline handling time on parallel execution with an exactness of 90.6 %. At last, in regards to Q3, the outcomes demonstrate that the added substance metric predicts reconfiguration time with a normal exactness of 90.5 % and 87.6 % on the first and refined model separately.

We watch that the added substance measurements is more precise for edge preparing time than for reconfiguration time, and for the last mentioned, it is better in the easiest model. We trust this is because of the many-sided quality of the basic test. For the previous, we just assessed 14 situations (legitimate setups) including the reaction time of 8 parts. Additionally, preparing video outlines requires more CPU (and GPU) operations, whose execution time estimation is more exact, than info/yield operations. For reconfiguration time, we assessed 210 situations (moves among substantial setups) and stacking shared libraries required extra information/yield operations that influenced the exactness of estimations. Especially, the explore different avenues regarding the refined model is more mind boggling since it deals with the 10

Open CV libraries autonomously, when contrasted with the improved model that heaps and empties all libraries immediately.

From the straight relapse examination, we watch that the Pearson connection coefficient is near 1. For both edge preparing and reconfiguration time, the assessed qualities are somewhat higher than the deliberate ones by and large. At first sight, the inverse ought to be relied upon because of the overheads presented by the Component Manager for controlling segment execution and reconfiguring the handling chain. Nonetheless, these overheads are immaterial as for part reaction, startup and shutdown time. The Camera Reader segment makes one picture protest for each casing, while Image Window obliterates them, in this way, for the successive handling of video edges; the overhead presented by the CM just includes summoning every segment all together with a reference to the picture outline objects. For parallel handling, we anticipated that an overhead due would the synchronization of strings on the FIFO lines. Then again, reconfiguration requires repeating and parsing a rundown of operations trained by the Configuration Adapter module. These overhead circumstances are not considered in the examination show, since they are requests of greatness lower than execution properties values in Table 2.

The purpose behind these deviations is highlight associations, i.e., the determination of one component impacts non-practical properties of different elements. The effect of these connections is better watched when assessing outline handling time. The reaction time of segments in the errand subordinate stage shifts relying upon the picture yield of the segment in the channel arrange. These segments apply some picture preprocessing channels that change picture attributes thus diminishing the reaction time of the accompanying parts. For example, the designs utilizing Face Detection with and without Image Smoothing have a normal preparing time of 252.78 ms and 306.81 ms separately in parallel execution, what demonstrates that smoothing decreases confront identification reaction time extensively.

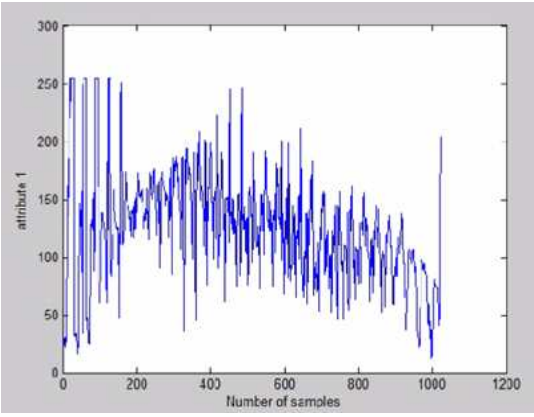

Figure 4 : With 1 Attribute

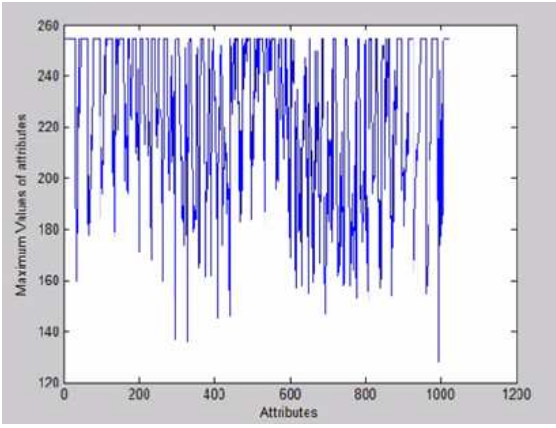

Figure 5 : Maximum Values of Attributes

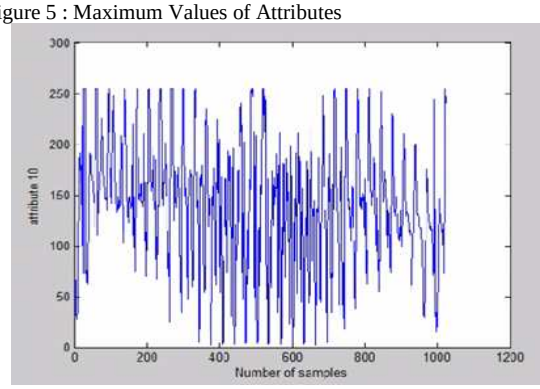

Figure 6 : With 10 Attributes

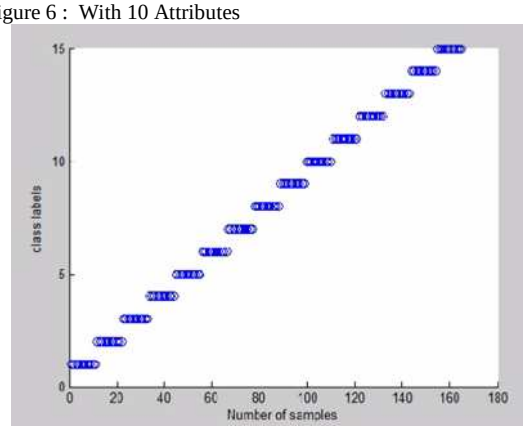

Figure 7: Attribute Representation

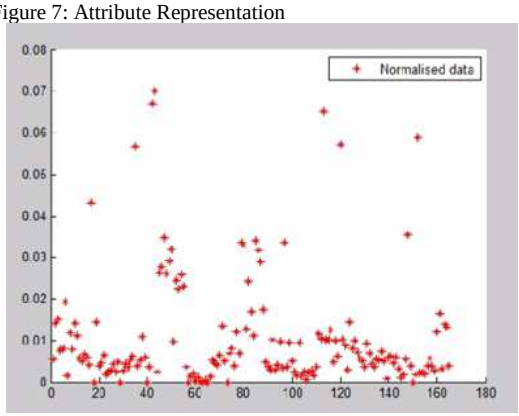

Figure 8: Normalised data

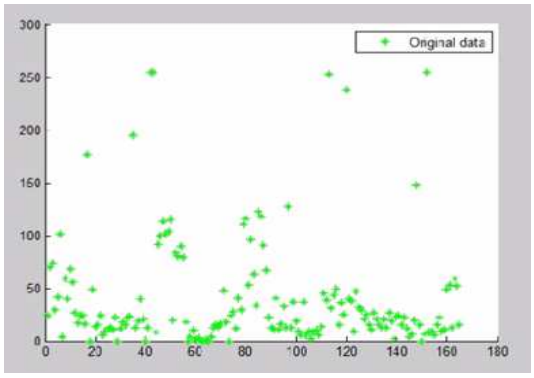

Figure 9 : Original Data

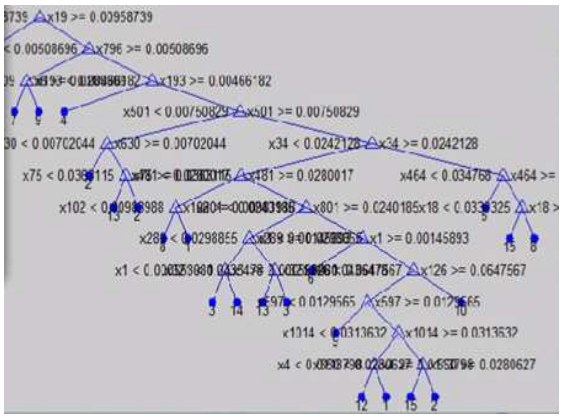

Figure 10: Tree Construction

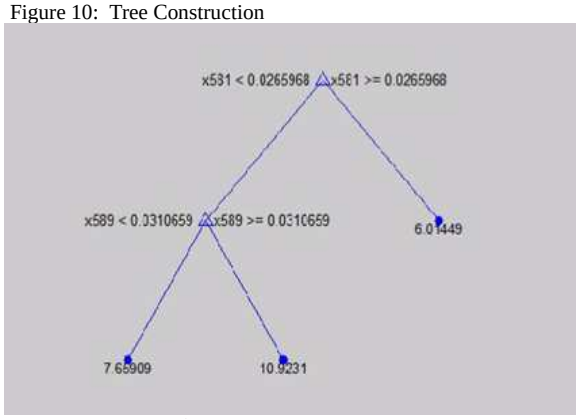

Figure 11: Output of Tree Construction

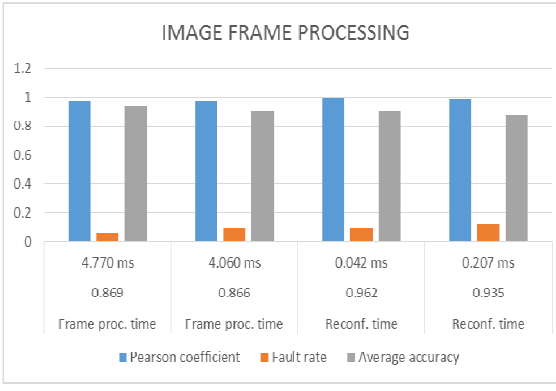

Figure 12 : Image Frame Processing

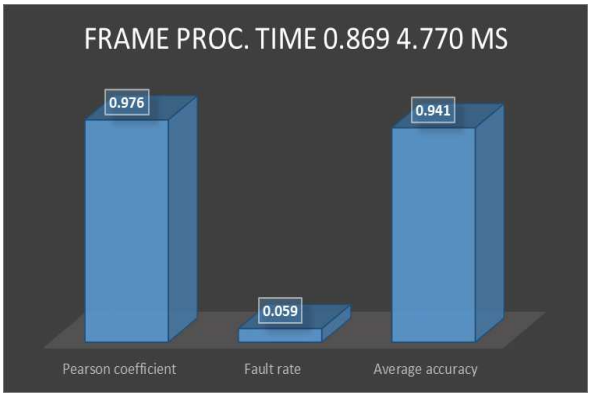

Figure 13 : sequential execution

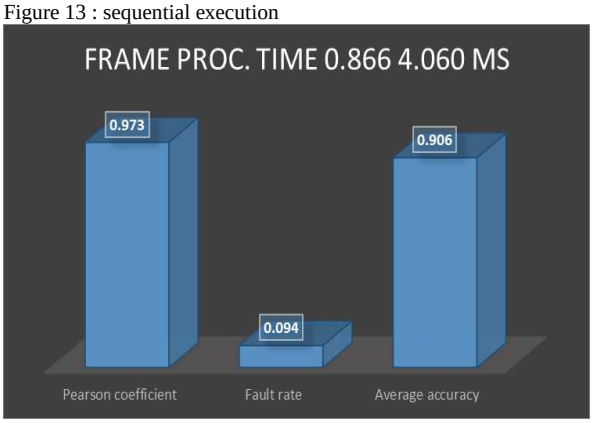

Figure 14 : parallel execution

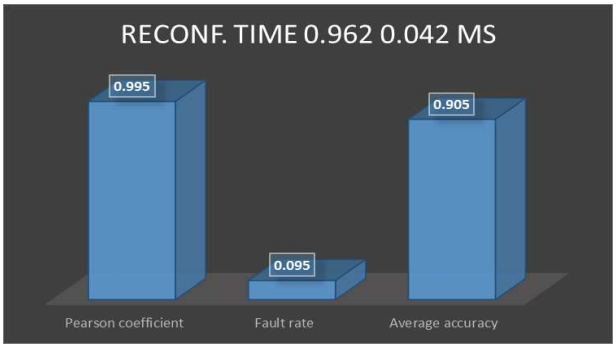

Figure15 : Pre loaded

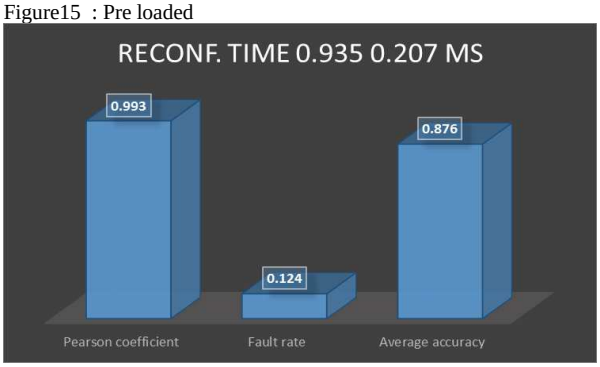

Figure16 : On Demand

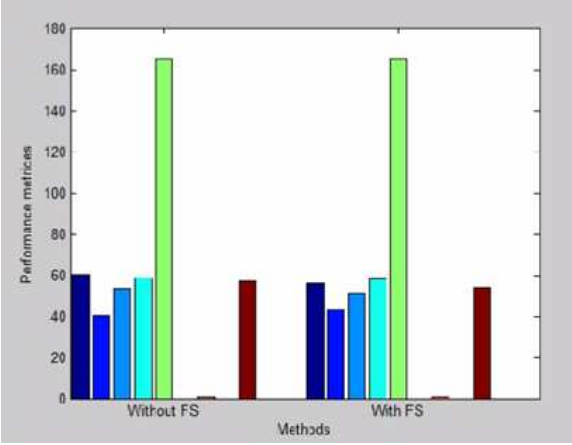

Figure 17 : Comparision of Feature Extraction

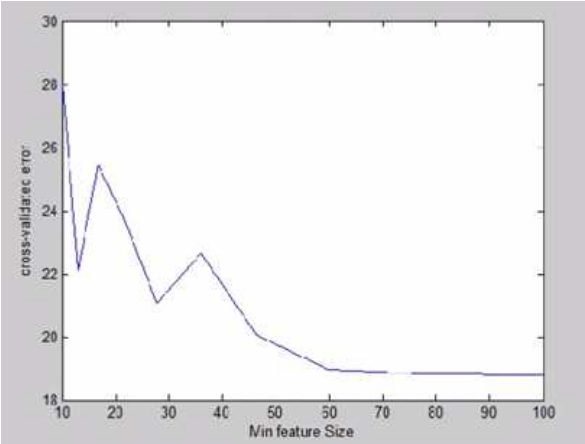

Figure 18: Feature Extraction

VI. CONCLUSION

This exploration enhances the dynamic adjustment of segment based frameworks by managing runtime quality properties on highlight models. The proposed structure quantitatively assesses and adjust numerous quality ascribes to achieve a superior framework arrangement. By incorporating part based stage that gives instruments to occasion taking care of and self-adjustment is done utilizing a more complex Component Manager usage. The approach is to embrace the space of administration arranged processing for helping the improvement, creation and mix of PC vision applications. In a conveyed setting, quality properties, for example, security, accessibility, versatility and battery utilization can be harder to oversee than in unified frameworks.

We examine approach restrictions and extend its materialness to other building styles. The setup determination is upheld by a heuristic inquiry calculation that guarantees rightness and fulfillment while tending to time proficiency and versatility for expansive scale occasions of the issue. , optimality can be accomplished at cost of a lower execution. The total capacities that we have considered are fitting for a few properties that are straightforwardly gotten from properties of individual parts. In future more assessments should be possible to concentrate the viability of these capacities and the impact of highlight collaborations on bigger models. Moreover, different measurements and runtime properties should be surveyed.

REFERENCES

[1] Benavides D, Segura S, Ruiz-Cortés a (2010) Automated analysis of feature models 20 years later: A literature review. Inform Syst 35(6):615–636

[2] Silva da DC, Lopes AB, Pinto FAP, Leite JC (2012) Selecting architecture configurations in self- adaptive systems using qos criteria. In: Sixth Brazilian Symposium on Software Components Architectures and Reuse (SBCARS), 2012. IEEE Computer Society, Washington DC, USA. pp 71–80. http://dblp.uni-trier.de/db/conf/sbcars/sbcars2012.html# SilvaLPL12

[3] Marler RT, Arora J (2010) The weighted sum method for multi-objective optimization: new insights. Struct Multidisciplinary Optimization 41(6):853–862

[4] Moisan S, Rigault JP, Acher M, Collet P, Lahire P (2011) Run time adaptation of video-surveillance systems: A software modeling approach. In: Crowley J, Draper B, Thonnat M (eds). Computer Vision Systems. Lecture Notes in Computer Science. Springer, Sophia Antipolis, France Vol. 6962. pp 203–212

[5] Morin B, File2ais O, Jezequel J, Fleurey F, Solberg A (2009) Models@ run.time to support dynamic adaptation. Computer 42(10):44–51

[6] Oliveira N, File2bosa LS (2014) A Self-adaptation Strategy for Service-based Architectures. In: VIII Brazilian Symposium on Software Components, Architectures and Reuse. SBCARS'2014. IEEE, Maceió, Alagoas Vol. 2. pp 44–53

[7] Pearl J (1984) Heuristics - Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley series in artificial intelligence, pp –1382. Addison-Wesley

[8] Rocha LM, Moisan S, Rigault JP, Sagar S (2011) Towards lightweight dynamic adaptation a framework and its evaluation.

[9] Sanchez LE, Diaz-Pace JA, Zunino A, Moisan S, Rigault JP (2014) An approach for managing quality attributes at runtime using feature models. In: VIII Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software SBCARS

[10] Siegmund N, Rosenmüller M, Kuhlemann M, Kästner C, Apel S, Saake G (2012) Spl conqueror: Toward optimization of non-functional properties in software product lines. Softw Quality J 20:487–517

[11] Siegmund N, Rosenmüller M, Kästner C, Giarrusso PG, Apel S, Kolesnikov SS (2013) Scalable prediction of non-functional properties in software product lines: File1tprint and memory consumption. Inform Softw Technol 55(3):491–507. Special Issue on Software Reuse and Product Lines Special Issue on Software Reuse and Product Lines

[12] Tran, V., Hummel, B., Liu, D.-B., Le, T., Doan, J.: Understanding and managing the relationship between requirement changes and product constraints in component-based software projects. In : The Thirty-first Hawaii International Conference On SystemSciences, Kohala Coast, HI , USA, pp.132 - 142 (1998)

[13] Dijkstra, E.: On the role of scientific thought. In : Edsger W. Dijkstra, Selected Writings on Computing: A Personal Perspective. Springer-Verlag, New York (1982) 60–66 36. Greer, D.: The Art of Separation of Concerns. (Accessed March 10, 2011)

[14] Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice 2nd edn. AddisonWesley Professional (2003)

[15] Feljan, J., Lednicki, L., Maras, J., Petricic, A., Crnkovic, I.: DICES technical report Classification and survey of component models. Technical report (2009) http://jeffreypoulin.info/Papers/ICSR94/icsr94.pdf , 10-05.2012

[16] WILLIAM FRAKES; CAROL TERRY, "Software Reuse: Metrics and Models" Available: http://www.cin.ufpe.br/~in1045/papers/art06.pdf , 10-05.2012

[17] Nasib S. Gill," Reusability Issues in Component-Based Development", ACM SIGSOFT Software Engineering Notes Volume 28 Issue 4, July 2003, doi:10.1145/882240.882255.

[18] Marcus A. Rothenberger; Derek Nazareth, A cost-benefit-model for systematic software reuse [online]. Available: http://is2.lse.ac.uk/asp/aspecis/20020086.pdf , 10-05.2012

[19] Xia Cai et el, "Component-based software engineering: technologies, development frameworks, and quality assurance schemes", APSEC '00 Proceedings of the Seventh Asia-Pacific Software Engineering Conference, IEEE Computer Society Washington, DC, USA ©2000. Available:http://www.cse.cuhk.edu.hk/~lyu/paper_pdf/apsec.pdf, 10-05.2012.

[20] Jasmine K.S; Dr. R. Vasantha, "Cost Estimation Model For Reuse Based Software Products", Proceedings of the International

MultiConference of Engineers and Computer Scientists 2008 Vol I, IMECS 2008, 19-21 March, 2008, Hong Kong.

[21] Ruben Prieto-Diaz, "DOMAIN analysis: an introduction", the Contel technology center acm Sig soft software Engineering notes vol 15 no 2.