Global Journal of Programming Languages

Volume No. 8

Issue No. 1

January - April 2023

ENRICHED PUBLICATIONS PVT. LTD

S-9, IInd FLOOR, MLU POCKET,
MANISH ABHINAV PLAZA-II, ABOVE FEDERAL BANK,
PLOT NO-5, SECTOR-5, DWARKA, NEW DELHI, INDIA-110075,
PHONE: - + (91)-(11)-47026006

Global Journal of Programming Languages

Aims and Scope

Global Journal of Programming Languages is a peer-reviewed Print + Online journal of Enriched Publications to disseminate the ideas and research findings related to all sub-areas of programming languages. It also intends to promote interdisciplinary researches and studies in computer science maintaining the standard of scientific excellence. This journal provides the platform to the scholars, researchers, and PHD Guides and Students from India and abroad to adduce and discuss current issues in the field of programming and Computer Sciences.

Global Journal of Programming Languages

Global Journal of Programming Languages

(Volume No. 8, Issue No. 1, January-April 2023)

Contents

Cloud Computing Environment in Data Management: A New Paradigm of Information Technology Management

FaizAkram Dr. Rajeev Kumar
Department of Information Technology, Shri Venkateshwara University,
Gajraula (Amroha), U.P. India
Faculty of Engineering and Technology, Shri Venkateshwara University,
Gajraula (Amroha), U.P. India

ABSTRACT

The Cloud has turned into another vehicle for conveying assets, for example, computing and capacity to customers on request. As opposed to being another Technology in itself, the cloud is another plan of action wrapped around new advances, for example, server Virtualization that exploit economies of scale and multi-tenure to lessen the cost of utilizing data Technology assets. From one viewpoint, cloud computing is just the same old thing new in light of the fact that it utilizes methodologies, ideas, and best practices that have as of now been set up. The test of building predictable, accessible and adaptable data administration systems fit for serving petabytes of data for a huge number of users has gone up against the data administration inquire about group and in addition huge web ventures. The utilization of cloud computing is getting to be distinctly across the board; however systematic investigation of its administrative ramifications is deficient. This paper looks at cloud computing with regards to other real changes in Data Technology (IT) and investigates the progressive changes and difficulties it conveys to data administration. The paper investigates the IT pendulum of centralization and decentralization and examines the administrative ramifications of the significant parts of cloud computing.

Keywords: Computing, Information Technology Management, centralization, decentralization, hardware, services,applications, virtualization

I. INTRODUCTION

The utilization of cloud computing is quickly developing, as is the writing on the specialized issues of execution. Our knowledge into the authoritative repercussions of distributed computing, regardless, still falls far behind. This paper inspects the wonder of cloud computing, places it With regards to other real changes in Information Technology (IT) and investigates the possibly progressive changes and difficulties it conveys to administration [1].

It is changing the remain solitary IT foundations to nearly look like social open Frameworks like power and water utility frameworks, and urging a move to on-demand and pay-per-utilize blueprints. For little to medium measured associations, this is starting at now streamlining IT limits, giving higher efficiencies while reducing General IT system and organization costs. Distributed computing will over the long haul diminish the prisoner IT impression of associations, streamline corporate IT organization, and change IT related expenses from tremendous frank capital costs (and advancing backing) to pay-as-you-use go approaches.

Cloud computing reception keeps on picking up energy over a wide scope of ventures including data administration. When associations figure out how to channel through the clamor encompassing the cloud, there are entirely exceptionally down to business courses in which the IT associations of banks, safety net providers and comparable organizations can use cloud computing to straightforwardly profit their day by day operations and most fundamentally, affect the business primary concern. These additions can be accomplished without causing huge capital use or uncovering delicate business data [2].

II. TYPES OF CLOUD MODEL

Public Clouds

Open or Outer cloud depicts cloud computing in the customary standard sense, whereby assets are progressively provisioned on a fine-grained, self-benefit premise over the Web, by means of web application/web services, from an off- webpage outsider supplier who impart assets and bills to a fine-grained utility computing premise. In this model, sellers progressively dispense assets (hard drive space, Smash, and processor control) on a for each client premise through web applications [3].

A couple of open cloud offerings have as of now turn out to be such an imbued part of the business group, for example, Cisco's WebEx meeting space and Salesforce.com Deals Cloud. Cisco and Salesforce.com aren't the main realmerchants to hop in with an open cloud offering - WebEx is joined by the Amazon Versatile Process Cloud (EC2), Google Applications, and Microsoft Sky blue.

Private Clouds

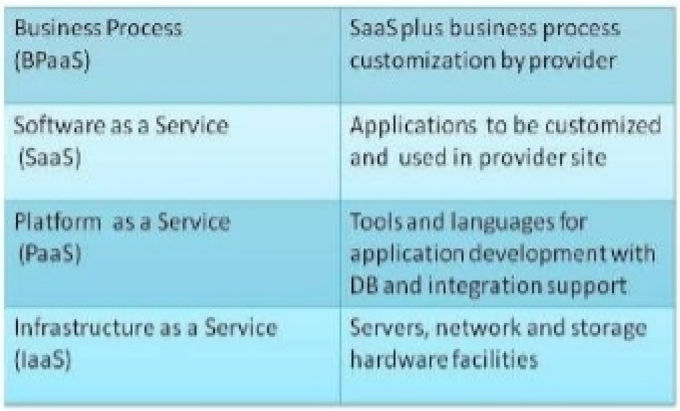
Private clouds alleviate these worries, with the security of an inner system. Since the client possesses the majority of the hardware fueling the cloud environment (frequently a huge data focus), the client has finish control over the IT assets and additionally the data and is in charge of securing it. In a private cloud, undertaking IT assets are merged and pooled so users over the organization can have self-benefit get to and expanded versatility. Likewise like an open cloud, a private cloud additionally makes arrangement a robotized benefit ask for as opposed to a manual errand prepared by IT [4].

Not at all like an open cloud, has setting up shop in a private cloud required aptitude with system coordination and with modern virtualization and cloud stage innovations. The association should run possess hardware, stockpiling, systems administration, hypervisor, and cloud programming.

Half breed Clouds

Mutt mists use a blend of internal resources, which stay under the control of the customer, and external resources passed on by a cloud master association. Like the private model, a mixture cloud gives an association a chance to keep on using their current data focus gear and keep touchy data secured by the association's own system. Besides, to general society cloud, a cream demonstrate allow a relationship to misuse a cloud's for all intents and purposes vast adaptability. It's an approach to illuminate a portion of the trust issues of the public cloud while getting the public cloud's benefits. Amazon's Virtual Private Cloud (VPC) is one of the primary instances of a cream cloud. With VPC, an association can in like manner expand its wellbeing endeavors, for instance, firewalls and interference revelation systems, to its AWS resources in the cloud [5].

Figure 1: Cloud Computing Service Models



Business Process (BPaaS)	SaaS plus business process customization by provider
Software as a Service (SaaS)	Applications to be customized and used in provider site
Platform as a Service (PaaS)	Tools and languages for application development with DB and integration support
Infrastructure as a Service (IaaS)	Servers, network and storage hardware facilities

III. INFORMATION TECHNOLOGY PHASES

Keeping in mind the end goal to better see how cloud computing fits in the pendulum of centralization and decentralization of Data Innovation, we ought to quickly inspect some real periods in the most recent four many years of advancement of IT in associations.

The primary time frame was the 1970's time of centralized servers and clump exchange preparing. IT was completely concentrated, and exchanges identified with finance, money related articulations, charging, bookkeeping systems and others were prepared in groups on the centralized server, disconnected, with the end-users basically accepting the yields [6].

The second time frame began in the 1980s, as exchange preparing moved to web based handling (e.g., Visas, ATMs, online reservation systems). Purpose of-Administration (POS) terminals got to be distinctly pervasive and EDI utilize (electronic data trade) got to be distinctly across the board. Amid this period, exchanges were still concentrated and still performed on the centralized computer, with the distinction that the accommodation interface was currently on the web and users could communicate specifically with the system by performing inquiries and getting reports.

The third time frame occurred in the 1990s, with the PC Transformation, the blast of end- client computing and inside business decentralization. Users put away data and ran applications all alone desktops or on their organization's system. At first they did all their computing at work, yet in the end home computing tagged along, and users could utilize their home PCs to run basic applications like word preparing and spreadsheets and perform little exchanges [7].

By the mid 1990's, organizations begun to get a handle on the IT capability of the Internet, yet auxiliary and specialized obstacles still stayed before they could completely use this potential. In the late 1990's, in any case, capital markets got the IT fever. Financial speculators got to be distinctly anxious to spend on IT, notwithstanding when the long haul way to gainfulness was not clear. This prompted to the burst of the theoretical rise in the mid 2000s, beginning a descending winding in IT that kept going until around 2003.

The Web 1.0 spoke to the fourth time frame in IT advancement, carrying mass decentralization and giving everyone with Web get to the ability to direct individual and work practices on the web: email, home keeping cash, electronic shopping, social coordinated effort, etc. The fifth time frame was the blend of Web 1.0 and outsourcing. The front end of business moved to the web, while the back end was outsourced– i.e., non-vital exchange handling systems, web bolster, anything that could be commoditized and done somewhere else on the planet at a lower cost, began being viewed as "services" that could be purchased from outside suppliers who could be anyplace (on-shore in the US, close shore in spots like Mexico, Canada and Focal America, or seaward in nations like China, India and Brazil). Outsourcing of IT errands and PC services that could be unmistakably characterized and were not part of the key center business permitted associations to change high IT settled expenses into lower outsourced variable expenses. Overseeing IT outsourcing organizations or collusions turned into an extremely complex process. CIOs wound up in a position that was much more requesting than before: in addition to the fact that they were still in charge of the IT capacities that stayed in-house, however they additionally got to be distinctly in charge of arranging, controlling and managing the conveyance of the outsourced IT services, while didn't really having direct specialist over these assets. Measuring execution now included measuring both achievement and disappointment, and furthermore deciding the obligation regarding disappointment in a domain where blame dealing was normal amongst customers and outsourcers [8].

The 6th and latest period is the blend of Web 2.0 or more distributed computing? This implies going past outsourcing, in light of the fact that both the front end and a portion of the back end of business can be outsourced. Rather than virtual associations, we have virtualized associations, with groups found anyplace on the planet teaming up using web 2.0 devices, net PCs, portable Technology and distributed computing administrations.

IV. DATA MANAGEMENT IN CLOUD

Data management has always been a challenge — for individuals, for small businesses, for big enterprises, and particularly for large, decentralized organizations like higher education institutions. The discipline of data management is not new — it started way back as records management, when paper files and folders were the data collection medium of choice. Unfortunately, legacy approaches based on paper often remain in place today for campus Technology, even when converted to electronic data formats and processes [9]. Once data management structure is established and operating well, It is ready to take on the new frontier of data management in the cloud. For example if SAAS in clouds is being considered as how a common mid-level venture application move to a product as an administration (SaaS) offered by cloud, for instance, understudy data administration that contains touchy data. Today, the vast majority of the Organizations likely get data documents from outsider sellers containing profile data about planned understudies, including contact data, provincial statistic data, ethnicity or sexual orientation, and conceivably secondary school or exchange.

Data is put away at an entrusted have. Despite the fact that it may not appear to bode well for a cloud computing Host Organization to disregard the security of its customers and get to data without authorization, such plausibility makes some potential customers apprehensive. When all is said in done, moving data off premises builds the quantity of potential security chances, and proper precautionary measures must be made. Moreover, in spite of the fact that the name "cloud computing" gives the feeling that the computing and capacity assets are being conveyed from a heavenly area, the truth of the matter is, obviously, that the data is physically situated in a specific nation and is liable to neighborhood tenets and controls. Data is duplicated, frequently crosswise over expansive geographic separations Data accessibility and sturdiness is principal for cloud stockpiling suppliers. Inaccessibility can data administration in Market-oriented Cloud Computing be damaging

both to the bottom line by failing to hit targets set in service level agreements and to business reputation. Information accessibility and strength are ordinarily accomplished through under-the-spreads replication (i.e., Data is consequently duplicated without client impedance or solicitations). Extensive distributed computing suppliers with server farms spread all through the world can give abnormal amounts of adaptation to non-critical failure by reproducing information crosswise over substantial geographic separations. Amazon's S3 distributed storage benefit reproduces information crosswise over "areas" and "accessibility zones" so that information and applications can persevere even notwithstanding disappointments of a whole area. The client ought to be mindful so as to comprehend the points of interest of the replication conspire notwithstanding; for instance, Amazon's EBS (versatile square store) will just duplicate information inside a similar accessibility zone and is along these lines more inclined to disappointments. It portrays the reasonableness of moving the two biggest parts of the information administration advertise into the cloud: value-based information administration and diagnostic information administration [10].

Transactional data management

Value-based data administration alludes to the databases that back keeping money, carrier reservation, online internet business, and production network administration applications. These applications ordinarily depend on the Corrosive property. The value- based data administration applications are not liable to be sent in the cloud because of taking after reasons:

Value-based data administration systems don't utilize common nothing engineering. The value- based database market is ruled by Prophet, IBM DB2, Microsoft SQL Server, and Sybase. Of these four items, neither Microsoft SQL Server nor Sybase can be sent utilizing mutual nothing engineering. IBM discharged a mutual nothing execution of DB2 in the mid-1990s which is currently accessible as a "Database Dividing Highlight" (DPF) add-on to their lead item, yet is intended to help scale logical applications running on data distribution centers, not value- based data administration. Prophet had no mutual nothing usage, however once more, this execution is planned just to be utilized for data stockrooms [11]. Executing a value-based database system utilizing mutual nothing engineering is non-inconsequential, since data is apportioned crosswise over locales and, by and large, exchanges can't be confined to getting to data from a solitary site. This outcomes in complex appropriated bolting and submit conventions, and in data being dispatched over the system prompting to expanded dormancy and potential system transmission capacity bottlenecks. Besides the principle advantage of a common nothing engineering is its adaptability; however this preferred standpoint is less pertinent for value-based data handling for which the larger part of organizations are under 1 TB in size. It is difficult to keep up Corrosive security in the circumstance where data replication over extensive geographic separations. The Top hypothesis [10] demonstrates that a mutual data system can just pick at most two out of three properties: consistency, accessibility, and resistance to parcels.

V. MANAGERIAL IMPLICATIONS

In the first period of IT evolution we examined here (mainframes and batch transaction processing, fully centralized IT, end-users receiving outputs), computers existed in a “secret world” separate from users, who were not familiar with them as physical objects, nor with their operations and jargon. During the second period (centralized servers and online exchange preparing, despite everything it concentrated), Pcs turned out to be to a greater extent an obvious substance, as end-users began cooperating with them through interfaces, for example, ATMs and online reservation systems; the nature of working together was changed by IT, yet that change did not achieve administration, who could at present consider that Technology was another person's issue. This administration protection changed in the third time frame (PCs, end- client computing, EDI), with interior business decentralization and administration's acknowledgment that they were currently in charge of overseeing their own particular association, as well as a system of between authoritative connections and organizations with customers and providers.

Amid the fourth time frame (Web 1.0, mass decentralization and full access to email, home keeping money, web based shopping, social cooperation, and so forth.), the web significantly lessened the expenses of EDI-like organizations, making it workable for organizations of all sizes to have a wide web nearness. Large portions of them, be that as it may, in any case looked after "dividers" between their on the web and physical operations, and needed to take in some hard lessons as they climbed the expectation to learn and adapt of speculation as a consistent association [12].

In the fifth time frame (Web 1.0 or more outsourcing), the front end of the business moved to the web, with the commoditization and outsourcing of non-centered trade get ready frameworks and support. As customers grabbed the ability to use social electronic media in every piece of their lives, web business got an amazing main thrust, and both private endeavors and broad associations got the chance to be locals of the web.

The 6th period (Web 2.0 or more cloud computing) is opening a time of basic administrative changes of business associations, with virtualized associations utilizing web 2.0 devices, net PCs, versatile Technology and cloud computing services.

As opposed to creating power from their own individual generators, associations got the chance to be particularly prepared to buy control from electrical utilities, which both cut costs and improved relentless quality.In such a world, By a similar token, cloud computing frees organizations from generating and deal with their own computing power, liberates them from the centralized computer and desktop- driven systems of the past and opens a future where they can expect all inclusive, day in and day out access to computing assets that another person is giving and overseeing in the cloud. In such a world, virtualized associations depend on groups that utilization Web 2.0 and the cloud to work together anyplace, at whatever time. This is not simply IT change, but rather a potential administration upset. With the cloud "the world movements from utilizing Technology (IT) for exchange and Technology administration to a much more natural Business Technology (BT) for joint effort and cooperation administration."

VI. CONCLUSION

It is reasoned that value-based data administration applications are not appropriate for cloud organization. The attributes of the data and workloads of normal explanatory data administration applications are appropriate for cloud arrangement. The flexible register and capacity asset accessibility of the cloud is effectively utilized by a mutual nothing design, while the security dangers can be to some degree lightened. Cloud computing changes the way IT experts will work, and the sorts of occupations they will have. Yet, it likewise gets a crucial change how chiefs consider business, organize undertakings and individuals.

It's about following up on circumstances, and giving others a chance to lead the pioneer when they know best about stuff being finished. In spite of the fact that the Cloud empowers radical change, the way of life of the firm will decide the result. Authorization, hazard resilience, developing bunches of little wagers – these are a portion of the reserves of a Cloud- situated business culture".

REFERENCES

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R, Konwinski, A., Lee, F., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M. "Over the mists: A Berkeley perspective of distributed computing." University of California Berkeley, Reliable Adaptive Distributed Systems Laboratory, Technical Report No.UCB/EECS-2009-28, (2009). Gotten to January 20, 2011 [available at http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html]

[2] Cagle, K. "Be that as it may, what precisely "is" distributed computing?" O'Reilly Broadcast, 2008. Gotten to January 20, 2011 [available at http://broadcast.oreilly.com/2008/12/yet what- precisely is-cloud-comp.html]

[3] Carr, N. The enormous switch: rewiring the world, from Edison to Google. W.W. Norton and Co., New York, NY, 2008.

[4] Fingar, P. "Distributed computing set to unleash an immaculate tempest in business." Cordial Cloudburst, 2009. Gotten to January 20, 2011 [available at http://www.cordys.com/ufc/file2/cordyscms_si tes/download/6f5f4d1cfe8be9d78d972fa808d8702c/ pu/c ordial_fingar.pdf]

[5] Gartner Research. "Meaning of Cloud Computing." In: Lori MacVittie, "Distributed computing: It's the goal, not the adventure that is critical." DevCentral Weblog, 2008. Gotten to January 20, 2011 [available at http://devcentral.f5.com/weblogs/macvittie/do cument/2008/11/03/distributed computing its- the-goal not-the-excursion that-is.aspx]

[6] Gillett, S. E. and Kapor, M. "The self- administering Internet: Coordination by Design." In: Coordination of the Internet, Brian Kahin and James Keller (Eds.). MIT Press, Cambridge, MA, 2008.

[7] Mell, P. and Grance, T. The NIST Definition of Cloud Computing. U.S. Bureau of Commerce, National Institute of Standards and Technology, Technology Laboratory.Rendition 15, 10-7-09, 2009. Gotten to January 20, 2011 [available at http://csrc.nist.gov/bunches/SNS/distributed computing/cloud-def-v15.doc]

[8] Miller, M. Distributed computing: Web- based applications that change the way you work and team up on the web. Que Publishing, Indianapolis, IN, 2008.

[9] Ommeren, E. V., Duivestein, S., deVadoss, J, Reijnen, C. and Gunvaldson, E. Joint effort in the Cloud. Microsoft and Sogeti, Bariet, Ruinen, the Netherlands, 2009.

[10] Oram, A. "Distributed computing points of view and inquiries at the World Economic Forum," WikiContent, 2009. Gotten to January 20, 2011 [available at http://commons.oreilly.com/wiki/index.php?title=Cloud_computing_perspectives_and_questio ns&prin table=yes]

[11] Urquhart, J. "The cloud discussion is evolving." CNET News: The insight of mists, June 6, 2009. Gotten to January 20, 2011 [available at http://news.cnet.com/8301-19413_3- 10258721-240.html]

[12] Urquhart, J. "Are the Feds the first to a typical cloud definition? CNET News: the insight of mists, May 10, 2009. Gotten to January 20, 2011 [available at http://news.cnet.com/8301- 19413_3-10237274-240.html]

A Survey on use of Search based Optimization Techniques in Software Engineering

Sanjiv Sharma*, S. A. M. Rizvi **, VineetSharma***

* Department of Computer Science and Engineering,,KIET Group of Institutions, Ghaziabad , India

** Department of Computer Science, Jamia Millia Islamia, Delhi, India

*** Department of Computer Science and Engineering,KIET Group of Institutions, Ghaziabad , India

ABSTRACT

In recent years search based optimization techniques in software engineering has been a burgeoning interest among software engineering. The Search Based Optimization Techniques are used to shift problem of software optimization from human based search to machine based search, by using techniques like meta-heuristic search and evolutionary computation. The idea behind this paradigm is to mingle human creativity with computing machine's reliability. This article presents a survey on some good work already done in this field.

Keywords-search based software engineering; hill climbing; simulated annealing; evolutionary algorithms; testing

INTRODUCTION

The Search Based Optimization Techniques are used to shift problem of software optimization from human based search to machine based search, by using techniques like meta-heuristic search, evolutionary computation and operations research. These techniques try to combine human's creativity and machines reliability [1]. Search Based Software Engineering (SBSE) is the name of a field in which Search Based Optimization is applied to Software Engineering. In a search based problem optimal or near optimal solutions are hunted in a search space of candidate solutions. These solutions are guided by a fitness function that distinguishes between better and worse solutions [1]. The term SBSE was coined by Harman and Jones [2] in 2001, which discusses Search Based Optimization as a general approach to Software Engineering. SBSE has been used in many fields within the general area of Software Engineering, e.g., requirements, design and testing.

SBSE can be useful in finding out smallest set of test cases, best architecture of the system, set of requirements that optimizes development cost and customer satisfaction, optimal allocation of resources to software project and best sequence of refactoring steps [3]. The rest of the paper is organized as follows, Section 2 briefly discuss some of the search based optimization algorithms techniques. Section 3 discusses some applications of SBSE in software requirements and specifications. Section 4 presents some applications of SBSE in software design. Section 5 discusses use of SBSE in testing and last section is conclusion.

SEARCH BASED OPTIMIZATION ALGORITHMS

There are two important ingredients for application of search based optimization in software engineering problems. First is selection of the representation of the problem and second is definition of the fitness function. There are a lot of problems in software engineering that have software metrics associated with them. These metrics are good candidates for fitness function [3].With the help of these two ingredients it is possible to implement search based optimization algorithms. Every search based algorithm use different approach to find optimal or near to optimal solutions. Approach for finding of solution is based upon fitness function, because fitness function is used to compare candidate solutions.

A. Hill Climbing Algorithm:

This algorithm selects an initial candidate solution randomly. It then examines the other candidate solutions, which are present in the neighborhood of the initial solution. These candidate solutions are similar but differ in some aspect. If a neighboring candidate solution possesses better fitness value compared to current solution, the search moves to the new solution. Now algorithm explores neighborhood of this new solution for better solution, and so on, until there is no improvement on the current candidate solution. Solution obtained using this method is called locally optimal and may not represent globally optimal solution, so search is repeatedly restarted with different initial solution in order to find out best solution. Number of restart of algorithm is decided by computing resources and available time [4].

B. Simulated Annealing Algorithm

This algorithm was proposed by Kirkpatrick et al. [5], is variation of Hill Climbing algorithm that avoids the local maxima problem by allowing candidates solutions of poorer fitness value. The probability of acceptance p of an inferior solution changes during searching of solution, and is calculated as: $p=e-\delta/t$, where δ is the difference value between fitness value between the current solution and the neighboring inferior solution being considered, and t is the control parameter known as the temperature. Temperature is cooled according to some cooling schedule. Due to high temperature at initial stage free movement is available is available in search space and it leads to lesser dependency on starting solution. As the search progresses, however, the temperature reduces, making moves to poorer solutions more and more unlikely. Eventually, freezing point is reached, and from this point on the search behaves identically to Hill Climbing. The name “Simulated Annealing” originates from the analogy of the technique with the physical process of annealing: the cooling of a material in a heat bath. When a solid material is heated past its melting point, and then cooled back into a solid state, the structural properties of the final material depends on the rate of cooling.

C. Genetic Algorithms

These algorithms come under the category of global searches due to considering many candidates solutions in the search space at once. The set of candidate solutions currently under consideration is called current population and each successive population considered is referred as a generation. These algorithms are inspired by Darwinian Evolution, here; each candidate solution is represented as a vector of components referred to as individuals or chromosomes. Generally, a Genetic Algorithm uses a binary representation, i.e. candidate solutions are encoded as strings of 1s and 0s; however more natural representations to the problem may also be used, for example a list of floating point values. In Genetic Algorithms first generation is made up of randomly selected chromosomes. Each individual in the population is then evaluated for fitness. On the basis of fitness value, certain individuals are selected to go forward to the following stages of crossover, mutation and reinsertion into the next generation. In Holland's original Genetic Algorithm [6] fitness-proportionate selection was chosen. In this selection system, the expected number of times an individual is chosen for reproduction is proportionate to the individual's fitness in comparison with the rest of the population. Selection based on fitness value leads the search to prematurely converge. However Linear ranking [7] and Tournament selection [8] have been proposed to check these problems.

Once the set of parents has been selected, recombination takes places to form the next generation. Crossover is applied on individuals selected at random with a probability pcross (referred to as crossover probability). After crossover, the offspring are inserted into the new population. If crossover does not take place, the parents are simply copied into the new population. After recombination, mutation is done, which is responsible for introducing genetic material into the search, in order to maintain diversification. This is generally achieved by flipping bits of the binary strings at some low probability rate pmute (referred to as mutation probability), which is usually less than 0.01. The search is terminated when some stopping criterion has been met, for example when the number of generations has reached some pre- imposed limit.

I. REQUIREMENTS/SPECIFICATIONS

Requirements engineering is a elementary component of the Software Engineering process [9], to which SBSE has also been applied in order to optimize choices among requirements, the prioritization of requirements and the relationships among requirements and implementations. One main goal is to select near optimal subsets from all feasible requirements to satisfy the demands of the customers, while at the same time making sure that there are sufficient resources to undertake the selected tasks. In the NRP, the goal is to find the ideal set of requirements that balance customer requests, resource constraints, and requirement interdependencies is called Next Release Problem (NRP). NRP can be formulated as a search problem [10]. In this formulation NRP problem is considered as a single objective optimization problem. In [10] applied a variety of techniques to a set of synthetic data to demonstrate the feasibility of SBSE for this problem. An iterative Genetic Algorithm proposed by Greer and Ruhe [11] for NRP. This approach balances the resources required for all releases; assessing and optimizing the extent to which the ordering conflicts with stakeholder priorities. Two objectives cost to the provider and estimated satisfaction rating for the customer is considered in [12].

Among various advantages of SBSE in requirement phase is robustness in volatile requirements [13], insight [12], requirements prioritization [13] and fairness in requirements assignment.

There are also some challenges in applying Search based Requirements Optimization, such as scalability, solution representation, fitness function definition, algorithm selection and requirement dependencies [14].

II. SOFTWARE DESIGN

The center of every software system is its architecture. Designing software architecture is a challenging task requiring much proficiency and knowledge of different design alternatives, as well as the ability to understand high-level requirements and piece them to detailed architectural decisions. Search-based approaches have been used in architecture level. In order to enhance and predict software quality search-based methods with some suitable fitness function are used in designing phase.

Amoui et al. [15] use the GA approach to improve the reusability of software by applying architectural design patterns to a UML model and to figure out the best sequence of transformations. Chromosomes are used for encoding of a sequence of transformations and their parameters. Each individual consists of several supergenes, each of which represents a single transformation. A supergene is a group of neighboring genes on a chromosome which are closely dependent and are often functionally related. Mutation randomly selects a supergene and mutates an arbitrary number of genes, inside that supergene and after this, check out its validity. If a transformed design contradicts with object-oriented concepts, for example, a cyclic inheritance, a zero fitness value is assigned to chromosome. Two versions of crossover are used. First one is a single-point crossover for supergene level, with a arbitrarily selected crossover point, which swaps the supergenes beyond the crossover point. The second one crossover arbitrarily selects two supergenes from two parent chromosomes, and similarly applies single point crossover to the genes inside the supergenes. This combines the parameters of two successfully applied patterns. Quality of the transformed design is examined, as introduced in [16], by calculating its “distance from the main sequence” (D), which unite several object-oriented metrics by calculating abstract classes’ ratio and coupling between classes, and measures the overall reusability of a system.

Bowman et al. [17] discuss the use of a multi-objective genetic algorithm (MOGA) in solving the class responsibility assignment problem with the objective of optimization of the class structure of a system through the placement of methods and attributes. The strength Pareto approach is used, which differs from a traditional GA by containing records of individuals from past populations. In paper each chromosome is represented as an integer vector. Each gene represents a method or an attribute in the system and the integer value in a gene represent the class to which the method or attribute in that locus belongs. A separate matrix is used for storing of dependency information between methods and attributes. Mutations are performed by simply changing the class value arbitrarily; One-point one traditional crossover is used. The fitness function is formed of five different values measuring cohesion and coupling: 1. method-attribute coupling, 2. method-generalization coupling, 3. method-method coupling, 4. cohesive interaction and 5. ratio of cohesive interaction. Selection is made with a binary-tournament selection where the fitter individual is selected 90% of the time.

Kessentini et al. [18] consider the transformation mechanism as a combinatorial optimization problem with the goal of finding out a better transformation for a given small set of variable examples. In order to achieve the goal authors combine transformation blocks extracted from examples to generate a model named model transformation as optimization by example (MOTOE). Model is based upon an adapted version of particle swarm optimization (PSO).

In this adapted PSO transformation solutions are represented as particles that exchange transformation blocks in order to converge towards an optimal transformation. Paper also discusses about two main advantages of MOTOE, it recommend a transformation without deriving transformation rule first, and it can be operated independently from the source and target metamodels.

Räihä et al [19] proposed an approach for automation of synthesization of software architecture using genetic algorithms. This technique applies architectural pattern for mutation and quality metrics (modifiability and efficiency) for evaluation and produces a proposal for software architecture on the basis of functional requirement given as functional responsibilities in terms of a graph. The behavior of genetic synthesis process is analyzed with respect to the effect of dynamic mutation, quality improvement speed, and the effect of quality attribute prioritization. Research result shows that it is feasible to genetically synthesize architectures with high fitness value.

III. TESTING

Search-based software testing is the application of metaheuristic search techniques to generate software tests. The fitness function is transformation of test adequacy criterion and helps in finding out best solution in the search space. The application of metaheuristic search techniques for testing is useful, because exhaustive testing is infeasible for big size and complex softwares. Search-based software testing has been used across white-box (structural), black-box (functional) and grey-box (combination of structural and functional) testing.

Harman et al. [20] this paper presents a theoretical exploration of the global search technique embodied by Genetic Algorithms. After doing empirical study that compare the behavior of both global and local search based optimization on real world programs, it reveal that there exists of test data generation problem that suit each algorithm, thereby suggesting that a hybrid global- local search may be appropriate. The outcome of the study indicates that sophisticated search techniques, such as Evolutionary Testing can often be outperformed by far simpler search techniques. However, there exist test data generation scenarios for which the evolutionary approach is ideally suited. A further empirical study shows that in order to maximize coverage, Evolutionary Testing should be hybridized with the Hill Climbing approach.

Yano et al. [21] In this paper a new multi-objective implementation of the generalized extremal optimization (GEO) algorithm, named M-GEOvsl, is presented. It was developed to use as a test case generator to find transition paths from extended finite state machines (EFSM). This algorithm not only covers transition but also minimizes test lengths. M-GEOvsl can deal with variable length strings, and can generate solutions with different lengths. The steps of the algorithm are performed on general multi-objective problem in which the solution length is an element to be optimized. It must notice that although M-GEOvsl was developed in a specific context, it can in principle be applied to any multi-objective problem where the number of design variables is itself a variable of the problem. This paper has three main contributions, first is a dynamic approach used for generation of model-based test cases, in which the model is the artifact that is executed, instead of the implementation of system under test. Second is the transition paths, with the data that trigger them, in order to avoid infeasible path generation. Third is a multi-objective approach is proposed not only to cover the test purpose, but also to minimize the test case length. Fourth is dependence analysis is used to guide the search for solutions.

Assunção et al. [22] This paper presented MOCAITO (Multi-objective Optimization and Coupling-based Approach for the Integration and Test Order problem) approach for the integration and test order problem in varied software development contexts, where the units, components or classes can be components. The approach is instantiated in the object and aspect- oriented contexts, and evaluated with real systems and three algorithms: NSGA-II, SPEA2 and PAES. The algorithms are compared by using different number of objectives and four quality indicators. A dependency model is used to represent dependencies between units. In this paper ORD (Object Relation Diagram) model is used, and some most natural dependency relations between classes and between aspects and classes are considered. A cost model was also used after consideration of relevant information such as number of attributes, operations, and type and number of parameters of order cost. These two models are used as input to multi-objective optimization algorithms. The algorithms produce a set of beat solutions (good orders) that represent the best trade-off among the objectives. Now the tester selects an order according to the testing plans and environment, by using a priority rule. Results from the empirical evaluation of MOCAITO in OO (Object Oriented) and AO (Aspect Oriented) systems show that the three compared evolutionary algorithms can efficiently solve the problem.. However, based on the analysis of the quality indicators Generational Distance (GD), Inverse Generational Distance (IGD), Coverage (C) and Euclidean Distance (ED) from an ideal solution, NSGA-II appears more suitable in most cases considering all systems, with second and fourth objectives. The algorithm PAES, shows better performance for more complex systems. SPEA2 presents the greatest execution time for all systems.

Boussaa et al. [23] In this paper, a NS (Novelty Search) algorithm based on statement-coverage criterion for the test data generation problem has been introduced. In this approach, algorithm explores the search space by considering diversity as the objective function and try to optimize it. Author selects test cases based on a novelty score showing how different they are compared to all other solutions evaluated so far rather than a fitness-based selection. Using the NS approach is clearly a divergent evolutionary technique, inspired by natural evolution's drive to novelty that directly rewards novel behaviors instead of progress towards a fixed objective.

Srivastava et al. [24] In this paper a method for optimizing software testing efficiency is presented. This objective is achieved by identifying the most critical path clusters in a program. Author has developed variable length Genetic Algorithms in order to optimize and select the software weighted path clusters based on criticality. Due to infeasibility of exhaustive software testing author developed a selective approach to testing by selecting on those parts that are most critical so that these paths can be tested first. In other word testing efficiency can be increased by identifying the most critical paths.

Langdon et al. [25] Author have represented mutation testing as a multiobjective search problem in which the goal is to search for higher order mutants that are difficult to kill and syntactically similar to the original program under test. The approach uses higher order mutation testing, but also consider traditional mutation testing since a first order mutant may be a special case of a higher order. This approach has been implemented using a combination of Genetic Programming (GP), Genetic Algorithms and Monte Carlo sampling. The results reveal that the higher order GP mutation testing approach is able to kill higher order mutant (complex faults) of a real program.

Shamshiri et al. [26] In this paper, an empirical study on working of two approaches; Genetic algorithm and Random Search is done is done by using EvoSuite (Automatic Test Suite Generation for Java) unit test suite generator and selecting 1,000 classes randomly from the SF110 corpus of open source projects. Results reveal that there is little difference between the coverage achieved by test suites generated by evolutionary search compared to test suite generated using random search. An exhaustive analysis reveals that the genetic algorithm covers more branches of the type where standard fitness functions provide guidance.

IV. CONCLUSION

This paper has surveyed applications of search based optimization techniques in different phases of software development. There are a various kind of search based optimization techniques available such as searches based on a heuristic, genetic algorithms and evolutionary computation. Some technique performs well in one scenario while other works in some other scenario. Quality of result produced depends directly upon algorithm chosen, representation of the problem in hand and most importantly on fitness function designed. In survey it has been observed that use of SBSE is least in requirement/specification and substantial in testing while design phase comes in between.

REFERENCES

[1] Harman, Mark, et al. "Search based software engineering: Techniques, taxonomy, tutorial." Empirical software engineering and verification. Springer Berlin Heidelberg, 2012. 1- 59.

[2] Harman, Mark, and Bryan F. Jones. "Search-based software engineering."Information and software Technology 43.14 (2001): 833-839.

[3] Harman, Mark, S. Afshin Mansouri, and Yuanyuan Zhang. "Search based software engineering: A comprehensive analysis and review of trends techniques and applications." Department of Computer Science, King's College London, Tech. Rep. TR-09- 03 (2009).

[4] McMinn, Phil. "Search-based software test data generation: A survey."Software Testing Verification and Reliability 14.2 (2004): 105-156.

[5] S. Kirkpatrick, C. D. Gellat, and M. P. Vecchi. Optimization by simulated annealing. Science, 220(4598):671{680, 1983.

[6] J. H. Holland. Adaptation in Natural and Arti_cial Systems. University of Michigan Press, Ann Arbor, 1975.

[7] Darrell Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. D. Scha_er, editor, Proceedings of the International Conference on Genetic Algorithms, pages 116{121, San Mateo, California, USA, 1989. Morgan Kaufmann.

[8] K. Deb and D. Goldberg. A comparative analysis of selection schemes used in genetic algorithms. In Foundations of Genetic Algorithms, pages 69{93. Morgan Kaufmann, San Mateo, California, USA, 1991.

[9] Cheng, B., and J. Atlee. "From state of the art to the future of requirements engineering." Future of Software Engineering 2007 (2007).

[10] Bagnall, Anthony J., Victor J. Rayward-Smith, and Ian M. Whittley. "The next release problem." Information and software technology 43.14 (2001): 883-890.

[11] Des Greer and G¨unther Ruhe. Software release planning: an evolutionary and iterative approach. Information & Software Technology, 46(4):243–253, 2004.

[12] Yuanyuan Zhang, Mark Harman, and S. Afshin Mansouri. The multi-objective next release problem. In GECCO'07: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1129–1136. ACM Press, 2007.

[13] Mark Harman, Stephen Swift, and Kiarash Mahdavi. An empirical study of the robustness of two module clustering fitness functions. In ACM Genetic and Evolutionary Computation Conference (GECCO 2005), Washington, D.C., USA, June 25-29 2005.
[14] Zhang, Yuanyuan, Anthony Finkelstein, and Mark Harman. "Search based requirements optimisation: Existing work and challenges." Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg, 2008. 88-94.
[15] Amoui, Mehdi, et al. "A genetic algorithm approach to design evolution using design pattern transformation." International Journal of Information Technology and Intelligent Computing 1.2 (2006): 235-244.
[16] Martin, Robert C. "Design principles and design patterns." Object Mentor 1 (2000): 34.
[17] Bowman, Michael, Lionel C. Briand, and Yvan Labiche. "Solving the class responsibility assignment problem in object-oriented analysis with multi-objective genetic algorithms." Software Engineering, IEEE Transactions on36.6 (2010): 817-837.
[18] Kessentini, Marouane, Houari Sahraoui, and Mounir Boukadoum. "Model transformation as an optimization problem." Model Driven Engineering Languages and Systems. Springer Berlin Heidelberg, 2008. 159-173.
[19] Räihä, Outi, Kai Koskimies, and Erkki Mäkinen. "Genetic synthesis of software architecture." Simulated Evolution and Learning. Springer Berlin Heidelberg, 2008. 565-574.
[20] Harman, Mark, and Phil McMinn. "A theoretical and empirical study of search-based testing: Local, global, and hybrid search." Software Engineering, IEEE Transactions on 36.2 (2010): 226-247.
[21] Yano, Thaise, Eliane Martins, and Fabiano Luis De Sousa. "A multi-objective evolutionary algorithm to obtain test cases with variable lengths."Proceedings of the 13th annual conference on Genetic and evolutionary computation. ACM, 2011.
[22] Assunção, Wesley Klewerton Guez, et al. "A multi-objective optimization approach for the integration and test order problem." Information Sciences267 (2014): 119-139.
[23] Boussaa, Mohamed, et al. "A Novelty Search-based Test Data Generator for Object- oriented Programs." Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference. ACM, 2015.
[24] Srivastava, Praveen Ranjan, and Tai-hoon Kim. "Application of genetic algorithm in software testing." International Journal of software Engineering and its Applications 3.4 (2009): 87-96.
[25] Langdon, William B., Mark Harman, and Yue Jia. "Efficient multi-objective higher order mutation testing with genetic programming." Journal of systems and Software 83.12 (2010): 2416-2430.
[26] Shamshiri, Sina, et al. "Random or Genetic Algorithm Search for Object-Oriented Test Suite Generation?." Proceedings of the 2015 on Genetic and Evolutionary Computation Conference. ACM, 2015.

Comparative Study of MOSFET, CMOS & FINFET Base

Ramakant *, Raghvendra Singh**
* M.Tech. Scholar, EC Department, faculty of Engineering & Technology,
andhana Kanpur, Uttar Pradesh, India
** Assistant professor in department, electronic & communication engineering, Faculty of Engineering & Technology, Rama University, Kanpur, India

ABSTRACT

In system technology, FinFETs are performed many ways to manipulate leakage cutting-edge, dynamic current, quick channel effects (SCE) and put off over MOSFET and CMOS. FETs are promising substitutes to address quick channel effects (SCEs) higher than the traditional planar MOSFET's at deeply scaled generation nodes and accordingly allow persevered transistor scaling. on this paper, reviewed the comparative take a look at of FinFET with specific parameter related to Channel period, Leakage current, electricity and postpone over MOSFET and CMOS.

I INTRODUCTION

During the last three a long time, CMOS era scaling has been a primary driving force of the electronics enterprise and has supplied a course closer to both denser and faster integration. As the dimensions of the transistors is deceased, the thickness of the silicon dioxide gate dielectric has gradually reduced to boom the gate capacitance and thereby drive current, which gives upward push in tool overall performance. The CMOS scaling of the gadgets is facing demanding situations due to shrinking geometries, lower supply voltage, and higher frequencies, this have terrible effect at the device by means of growing quick channel effect due to which leakage (gate leakage and sub-threshold leakage) inside the device is growing constantly [9]. The enhancement in the scaling technology has increased the want of low energy primarily based circuits. In nanometer regime, CMOS based circuit are not be used due to problem in its fundamental cloth, short channel impact and high leakage. The higher technology are wanted for coping with the various effects of MOSFET era [1].because the planar MOSFETs indicates a great SCE (brief Channel impact) and the designers are pay attention to FinFETs, which have negligible SCE for the identical channel period [17]. FinFETs have attracted growing interest during the last decade due to the degrading brief-channel conduct of planar MOSFETs. at the same time as the planar MOSFET channel is horizontal, the FinFET channel (additionally referred to as the fin) is vertical. With a couple of fons and smaller fin heights ends in extra bendy and width of the channel may be increased, which in turn ends in more silicon place. on the other hand, taller fins result in less silicon footprint, but may bring about structural instability. although FinFETs carried out on SOI wafers are very popular. FinFETs have additionally been implemented on conventional bulk wafers appreciably. In [24] authors are finished their paintings on reduction of short channel effects in FinFET. The results screen that leakage cutting-edge due to DIBL is nicely suppressed and roll-off of a FinFET is nicely controlled. Authors in [9] designed a 6T SRAM cell using the FinFET and in comparison SRAM with conventional shape by using varying Sub-threshold leakage cutting-edge and gate leakage modern of inner transistors with self-controllable voltage stage technique. The simulation was carried the use of Cadence Virtuoso tool at 45nm generation and the simulation outcomes indicates that total leakage reduces up to 34% below the self-controllable voltage degree technique.

II. MOSFET devices

Metal Oxide Semiconductor FET may be very exceptional from that of the Junction FET. both the Depletion and Enhancement type MOSFETs use an electrical field produced with the aid of a gate voltage to alter the go with the flow of price providers, through the semi conductive drain- source channel. In fig 1, suggests that the gate electrode is positioned on the top of a very skinny insulating layer and there are a pair of small regions of n-kinds simply underneath the drain and source electrodes.

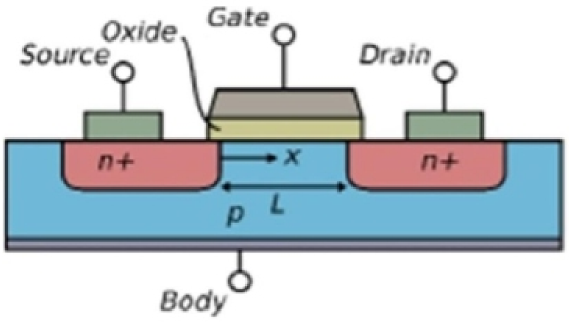


Fig 1: 2D view of MOSFET

In [26] authors have concluded that lowering the gate dielectric thickness increases the MOSFET drain modern or permits the deliver voltage to be decreased whilst retaining an acceptable drain current and enables the reduction of gate length through the suppression of short-channel outcomes. competitive scaling of CMOS era in latest years has reduced the silicon dioxide (SiO2) gate dielectric thickness. major reasons for difficulty in similarly reduction of the SiO2 thickness include extended poly-silicon (poly-Si) gate depletion, gate dopant penetration into the channel place, and excessive direct-tunneling gate-leakage current, which ends up in questions concerning dielectric integrity, reliability, and stand-with the aid of energy intake. As well identified in improved gate leakage modern in MOSFET is a assignment to continue in the scaling. In [18] authors display that minimize the MOSFETs to the nanometer regime ends in short channel results, which degrades the gadget overall performance and reliability.

III. CMOS

CMOS circuits are built in this type of way that every one PMOS transistors have an enter from the voltage supply or from any other PMOS transistor. similarly, all NMOS transistors have an enter from floor or from another NMOS transistor. Fig 2 suggests the composition of a PMOS transistor, when low voltage is carried out which creates a low resistance between its supply and drain and creates excessive resistance when a high gate voltage is applied. on the other hand, the composition of an NMOS transistor, while a low gate voltage is carried out creates excessive resistance between supply and drain and coffee resistance at a high gate voltage.

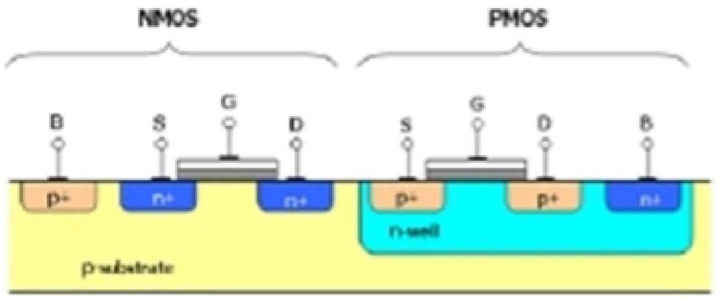


Fig 2: 2D view of CMOS

CMOS achieve modern-day reduction by way of emphasize their qualities of every nMOSFET with a pMOSFET and connecting each gates and both drains collectively. A excessive voltage on the gates, with a view to turn on the nMOSFET and pMOSFET will be in off nation, even as a low voltage at the gates will causes the reverse. This association significantly reduces energy intake and heat generation. but, at some point of the switching time of the gate voltage is going from one country to any other country, both MOSFETs behavior in short. This give rise to an abrupt spike in power consumption and arise a extreme issue at excessive frequencies.

IV. FINFET devices

Researchers are striving tough to broaden transistor with low price and high overall performance, one such development is a FinFET. It makes use of a fin like structure instead of the conventional flat layout, probable allowing engineers to create faster and more compact circuits and computer chips. The time period ―FINFET‖ describes a non-planar, double gate transistor built on an SOI substrate, primarily based at the unmarried gate transistor design. The crucial characteristics of FINFET is that the engaging in channel is wrapped by using a skinny Si ―fin‖, as shown in fig 3 which paperwork the frame of the device. The fin thickness, which determines the effective channel length of the device [7].

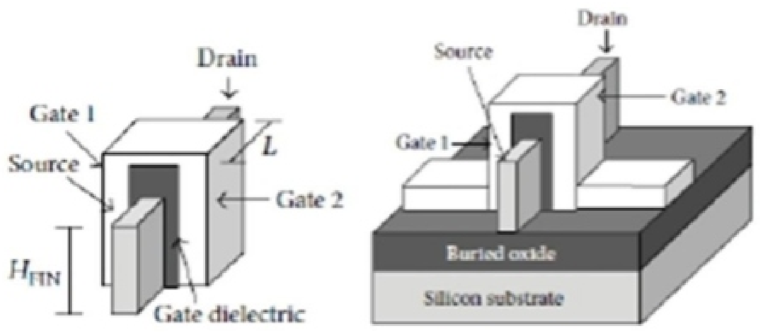


Fig 3:3D view of FinFET and SOI FinFET

The FinFET version structure consists of following regions With very low doping silicon fin area. Poly-silicon region, supply and drain contact location, exceedingly doped. Gate place- oxide (SiO2) tremendous control of brief channel results in submicron regime and making transistors nevertheless scalable. because of this reason, the small- duration transistor may have a bigger intrinsic advantage and much decrease off-kingdom present day in comparison to bulk counterpart. Promising matching behavior. Low value higher technological maturity than planar DG. Suppressed quick Channel impact(SCE) FinFET is one of the promising and higher technologies for its programs and the circuit layout for better overall performance and reliability. In [11] authors are evaluated the performance of FinFET based 6T SRAM mobile in 22nm era. The simulation result indicates that FINFET based totally SRAM design is feasible design in comparison with CMOS based totally SRAM design. effective Channel length (Leffg): maximum of the scaling conduct of FINFET isn't equal to poly period Lg which may be very widespread, but the effective channel length Leffg = Lg- ΔL. normally effective length of the transistor may be extracted from measurement or simulation result. but it isn't a dependable technique. The dependable approach is, by searching the full oxide capacitance Cox as a feature of Lg. The Cox is given by way of

$$Cox= (\varepsilon ox/tox)\ Weff\ (Lg-\Delta L) \qquad (1)$$

Gate Capacitance (Cg):- it's miles given with the aid of the relation

$$Cg = Co\ W\ L \qquad (2)$$

where C0 is the gate capacitance in step with unit region given by using

$$Co = \epsilon r\ \epsilon o / D \qquad (3)$$

wherein, D– Thickness of gate oxide, W – Width of the Channel L – length of the Channel Єr

Relative permittivity

Channel Resistance (RC): - it's far given by means of the relation

$$RC = \mu\ (L / W\ Qon) \qquad (4)$$

wherein,

$$Qon = Co \times Vgs \quad (4.a)$$

μ- Is carrier mobility

Vgs – gate to supply voltage

Gate delay (Td): - postpone of the gate may be calculated as

$$Td = Ron \ X \ Cg \quad (5)$$

Where in, Cg - Gate capacitance

The values of Gate capacitance, fee in keeping with unit location, Gate capacitance in step with unit region, Gate postpone and hannel resistance may be managed by way of adjusting the Oxide thickness [11].

impact on Leakage modern:

In present day CMOS technologies, the sub-threshold leakage current, ISUB, is a good deal large than the opposite leakage modern components. that is specially because of the fairly low threshold voltage in modern-day CMOS gadgets. ISUB is calculated by means of where k and n are capabilities of the generation. And η is the drain-brought about barrier reducing coefficient [21].

V. CHANNEL PERIOD:S

67

VI. LEAKAGE CONTEMPORARY AND DYNAMIC CUTTING-EDGE

revealed that the CMOS devices are shrinking to nanometer regime, growing the effects in brief channel results and versions inside the manner parameters which lead to reason the reliability of the circuit as well as performIn [13], authors have explored the blessings supplied by means of In [11], Authors have ance. The comparative outcomes became proven that FinFET based SRAM design is a possible layout in contrast with CMOS based design at 22nm era. In [21], they found an alternative for low strength interconnect synthesis on the 45nm node using Fin-type field- effect Transistors (FinFETs) which are a promising substitute for bulk CMOS at the considered gate lengths. The tested layout approach the output of FinFET SRAM became compared with general CMOS SRAM and the full-size outcomes were acquired that FinFET have more advantage than the CMOS one. In [18], the simulated consequences for FinFET based totally virtual software at nanometer had been acquired that the FinFET device are stepped forward giving higher results than the conventional MOSFET's. In [22], authors are presented the circuit design implication of Ge vs Si PMOS FinFET at the ten nm and seven nm nodes and extracted the internet list from the TCAD. And simulated consequences turned into confirmed that Si always out plays Ge across all critical circuits metrics at 10 nm node and therefore the Si answer remains preferred at 7 nm node. In [7], the fabrication of MOSFET, the minimal channel period were shrinking constantly. As gadgets reduce in addition and similarly, the problems with conventional (planar) MOSFETs are increasing. industry is currently at the 90nm node. As we cross down to the 65nm, 45nm, 32NM, 22nm, 14nm and many others., nodes, there appear to be no possible alternatives of persevering with forth with the traditional MOSFET. FinFET era gives an opportunity to classical MOSFET to reap low electricity packages and have negligible SCE for the same channel period. In [1], they in comparison the MOSFET with FinFET based totally logic circuit using 32 nm technology, and they found that among all the design FinFET based circuits are more suitable for low strength application in nanometer regime. in this section, Nanometer regime of FinFET are studied and it is discovered in most of the literature that in cutting down the era, FinFETs are appropriate substitute for planar MOSFET's.

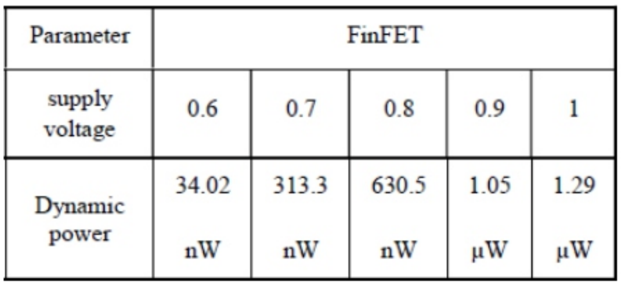
multi-gate fin FETs (FinFETs) over traditional bulk MOSFETs. The leakage-postpone saving changed into obtained with returned biasing were explored and evaluated the suitability for low wake-up time and electricity-efficient BB schemes. consequences have been confirmed that 3T and 4T FinFETs are decreased the leakage on the cost of a reasonably worse velocity performance and are well ideal for imposing fast and strength-green adaptive BB techniques. apparently, the better leakage discount performed via 4T FinFETs is because of the higher threshold sensitivity to back biasing became compared to bulk gadgets. In [20], authors are completed an extensive simulation on BOI FinFET and analyzed the effect of underlaps the usage of the TCAD. The simulation of BOI FinFET tool with underlap have been carried out. The incorporation of underlap with BOI FinFET device resulted in a enormous development in SCEs, especially DIBL reduced to 25%, leakage present day decreased to 99% and ION/IOFF extended to 89%.There might be discount in leakage modern-day and electricity dissipation, whilst device is in off condition. The simulations had been discovered that the BOI FinFET structures with underlaps are more efficient than conventional BOI FinFET as undoped underlap place reduces DIBL, leakage cutting-edge(IOFF) and advanced the ION/IOFF ratio. In [21] the leakage present day and dynamic contemporary are analyzed for the FinFET and CMOS primarily based 7T SRAM at 45 nm node. They taken into consideration a mechanism for improving FinFETs efficiency, known as variable-deliver voltage schemes. The transistor stacking at the side of variable deliver voltage operation of FinFETs obtained a larger leakage savings compared to that of bulk gadgets. The simulated outcomes for variant of supply voltage from minimum zero.6V to most of 1V are demonstrated that FinFETS save a larger leakage and dynamic present day. In [12], the leakage cutting-edge brought about by using DIBL was well suppressed. DIBL takes place while the depletion area of the drain interacts with the source close to the channel floor to lower the source ability barrier. DIBL is improved at better drain voltage and shorter powerful channel duration. floor DIBL is takes place before deep bulk punch-via and it decreased the brink voltage and the leakage cutting-edge. In [2] they designed 6T SRAM cellular the use of the FinFET and as compared it with conventional primarily based 6T SRAM at 45 nm node. The inner transistor sub-threshold leakage modern-day and gate leakage cutting- edge are determined and compared with the conventional structure of 6T SRAM cellular. FinFET SRAM cell is implemented with self-controllable voltage level method and then leakage modern and leakage strength additionally observed. The simulation results had been achieved on Cadence Virtuoso tool at 45nm era had a leakage contemporary in traditional SRAM mobile is 919.3 pA and that of SRAM is 858 pA. The Leakage in SRAM cellular is reduced to ten% the use of FinFET. The resulted general leakage of FinFET SRAM mobile is reduced to 34% once they carried out for self-controllable voltage level technique. We finish that higher reduction in leakage modern and dynamic contemporary is possible in FinFET technology as compared with conventional strategies.

VII. STRENGTH DISSIPATION

In [21], authors have analyzed for the dynamic electricity for FinFET and CMOS primarily based 7T SRAM at 45 nm node. The simulated results were showed that FinFET saves the more energy than that of CMOS. table I indicates the voltage version scheme for dynamic energy for FinFET at 45 nm node.

TABLE I: SIMULATION RESULTS OF FINFET 16 BIT SRAM ARRA

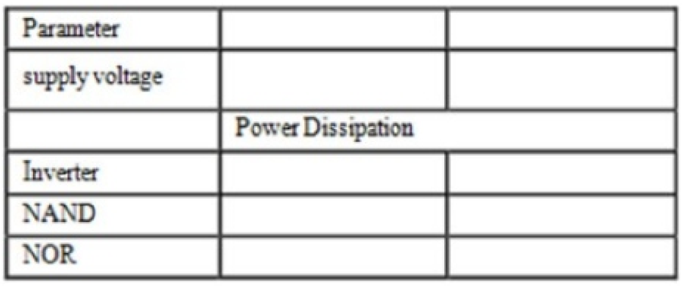
In [11] authors have evaluated and in comparison the performance of CMOS and FinFET primarily based 6T SRAM cell in 22 generation and compared the 6T SRAM for SG/IG mode are performed the dynamic power has 1.38e-04 and leakage power has 1.01e-06 at VDD of 0.8V. among FinFET primarily based



Parameter	FinFET				
supply voltage	0.6	0.7	0.8	0.9	1
Dynamic power	34.02 nW	313.3 nW	630.5 nW	1.05 µW	1.29 µW

TABLE I: SIMULATION RESULTS OF FINFET 16 BIT SRAM ARRAY

TABLE II: SIMULATION RESULTS FOR FINFET BASED INVERTER, NAND AND NOR



Parameter		
supply voltage		
	Power Dissipation	
Inverter		
NAND		
NOR		

SG and IG modes, SG mode design is giving better performance at all of the load stage. In [14], authors have analyzed the process triggered variations on 14-nm technology node silicon on insulator (SOI) FinFET and impact of these variations on postpone and static strength dissipation of FinFET inverter changed into offered. The fin width and peak versions had been taken into consideration that the less effect at the static power dissipation of inverter whereas fluctuation in gate oxide thickness has widespread effect on it. The gate oxide thickness is incremented of 10% which decreased the impact of Lg variability on static energy dissipation to eighty five.28%. The effect of version in static strength turned into minimized by way of deciding on thicker gate oxide as it turned into decreased the gate leakage modern. In [18] they curb the device from 22nm to 14 nm and compared the parameters in each nodes. The power consumption turned into decreased in 14 nm than that of twenty-two nm node. The table II indicates the compared power dissipation in 22 nm with 14 nm node. In [1], authors have compared the MOSFET and FinFET primarily based good judgment circuits the usage of 32nm generation and discovered that FinFET primarily based circuits are more appropriate for low strength applications in nanometer regime. The strength dissipation is discovered that 0.247µW for inverter and zero.131 µW and 0.096 µW for NAND2 gate, which is low in SG and IG mode. The FinFET based generation has much less energy consumption in comparison with traditional MOSFET and CMOS generation.

VIII. DELAY

In [14] authors have analyzed the FinFET logic circuits in terms of the postpone-tradeoff and in comparison with bulk circuits. They received the postpone saving with returned biasing and evaluated for low wake-up time and energy green BB scheme. In [eleven] they evaluated the performance have been analyzed for FinFET primarily based 6T SRAM and CMOS at 22 nm node. The end result of FinFET based totally 6T SRAM showed that the signal noise margin 000000(SNM) is multiplied to 25.3% in memory cell and it have become 5% faster than CMOS based totally SRAM. In [21]they taken into consideration the operation pace of FinFET SRAM and CMOS array. The examine and write delays are extracted from the characteristic of range of rows. It became observed that the FinFET realized the examine and write operation more or less two times fasterthan that of bulk planar one. desk III and IV indicates the in comparison postpone operation for CMOS and FinFET SRAM array

Parameter	CMOS 16 bit SRAM array				
Supply voltage	0.6	0.7	0.8	0.9	1
Delay (Write)	800.7ps	826.1ps	826.3 ps	818.5 ps	912.2 ps
Delay (Read)	575.4 ps	265.2 ps	771.1 ps	776.3 ps	820.3

TABLE III: SIMULATION RESULTS FOR CMOS 16 BIT SRAM ARRAY

Parameter	FinFET 16 bit SRAM array				
Supply voltage	0.6	0.7	0.8	0.9	1
Delay (Write ps)	200.2	188.3	174.5	182.1	193.7
Delay (Read ps)	425.8	254.2	229.6	224.6	229

In [14], authors have simulated the method caused variations on 14 nm technology node silicon on insulator (SOI) FinFET. The effect of process variant at the delay are minimized to 21.13 % and the height of the fin is reduced to 12%.In [18] they lessen the device from 22nm to 14 nm and in comparison the parameters in each nodes. The region covered by way of the tool is reduced. therefore the energy dissipation is also decreased but the delay inside the operation inside the device had been multiplied at 14nm node compared with 22nm. The desk V indicates the as compared effects for 22 nm and 14 nm. FinFET era have high velocity performance.

TABLE IV: SIMULATION RESULTS FOR FINFET 16 BIT SRAM ARRAY

Parameter	14nm	22nm
Supply Voltage	0.8V	0.9V
Inverter	74.0142ps	45.4125ps
NAND	74.2176ps	42.4052ps
NOR	86.2274ps	51.015ps

IX. CONCLUSION

In this paper, Process technology of FinFET is reviewed and the comparative study of MOS, CMOS and FinFET technologies is carried out. FinFET technology is a good alternative to planar CMOS for scaling beyond 32nm and has superior performance for low power applications.

REFERENCES

[1] Aditya waingankar et all.,2016,‖Logic Circuits using FinFET: Comparative Analysis‖ IJSRD, National Conference on Technological Advancement and Automatization in Engineering, January 2016 ISSN:2321-0613.

[2] Anshul Jain et al.,2014, ― Optimization of Leakage Current and Leakage Power of FinFET Based 6T SRAM Cell‖ Inte rnational Journal of Engineering Technology, Management and Applied Sciences, August 2014, Volume 2 Issue 3, ISSN 2349-4476

[3] Bazizi.E.M et al.,2015, ―Advanced TCADSimulation of Local Mismatch in 14nm CMOS Technology FinFET's‖ SISPAD 2015, September 9-11, 2015, Washington, DC, USA

[4] Bhattacharya D et al., 2014, —FinFETs:from device to architecture- Review Article‖, Received 4 June 2014; Accepted 23 July 2014; Published 7 September 2014.
[5] Dominique schreurs et al.,2010—Capabilities and Limitations of Equivalent Circuit Models for Modeling Advanced Si FET Devices‖, MIXDES 2010, 17th International Conference "Mixed Design of Integrated Circuits and Systems", June 24-26, 2010, Wroc_aw, Poland.
[6] Frank He et al.,2010, —FinFET: From Compact Modeling to Circuit Performance‖, 2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)
[7] Girish h et all.,2016, — Design of finfet‖ international journal of engineering research ISSN: 2319-6890) (online),2347-5013(print) volume no.5 issue: special 5, pp: 992-1128,20 may 2016
[8] Gracieli Posser et al.,2014, —Performance Improvement with Dedicated Transistor Sizing for MOSFET and FinFET Devices‖ Conference Paper • July 2014, DOI: 10.1109/ISVLSI.2014.13
[9] Jain A and Saxena M, 2014 —Optimization of Leakage Current and Leakage Power of FinFET Based 6T SRAM Cell‖ International Journal of Engineering Technology, Management and Applied Sciences, Vol:2, PP 156-163
[10] Khandelwal S et al.,2013 —Leakage current and Dynamic power analysis of FinFET based 7T SRAM at 45nm Technology‖, International Arab conference on Information Technology (ACIT 2013).

XHTML Transformation to JSF Component Tree

Vijay Kumar Pandey*
*Director of Technology Solutions, Intueor Consulting,
Inc. Irvine CA – USA, pandey@intueor.com

ABSTRACT

Java Server Faces (JSF) is a component-oriented framework to manage html-basedrequest and response. The current version of JSF 2.3 included as part of Java Enterprise Edition (JEE 8) has the default view technology of Facelet. Facelet is the java object representation for the XHTML (Extensible HyperText Markup Language) in the JSF context. Enterprise project teams building JSF based system typically write the view layer code in an XHTML format. XHTML are physical files,which are processed by the JSF run time engine to convert it into a runtime java object as Facelet and which are then transformed to component tree based on the various tag handlers associated with the xml tags present in the xhtml file. This paperlays down the ground work for the complex processing behind the transformation of xhtml to component tree which can help software architects and developers to better understand this processand helping them to design and maintain their JSF based system in an effective manner. This document assumes that the reader has a basic understanding of JSF.

Keywords: - XHTML, JSF Component Tree, VDL (View Declaration Language), CDI (Context Dependency Injection), Facelet, UIViewRoot, JSF Lifecycle.

I. INTRODUCTION

This papergoes in depth to describe the complex processing that occurs during the conversion of an XHTML file to fill up a JSF component tree UIViewRoot. XHTML files are converted to Facelet objects and then JSF component tree and for a http request this component tree resides in UIViewRoot.The default Facelet implementation provided by both Oracle's Mojarra&Apache's MyFaces is XHTML. This paper will lay out in detail the processing that occurs in the JSF runtime enginefor converting a physical XHTML file to a JSF component tree. The sample code provided in this paper utilizes open source projects such as PrimeFaces and OmniFaces. JSF Lifecycle consists of six phases, i.e., RESTORE_VIEW, APPLY_REQUEST_VALUES, PROCESS_VALIDATIONS, UPDATE_ MODEL_VALUES, INVOKE_APPLICATION, andPhases for XHTML to Facelet and Component Tree ProcessingRENDER_RESPONSE. The transfornation of XHTML to Facelet and component tree mainy happens in the RESTORE_VIEW and RENDER_RESPONSE phases.

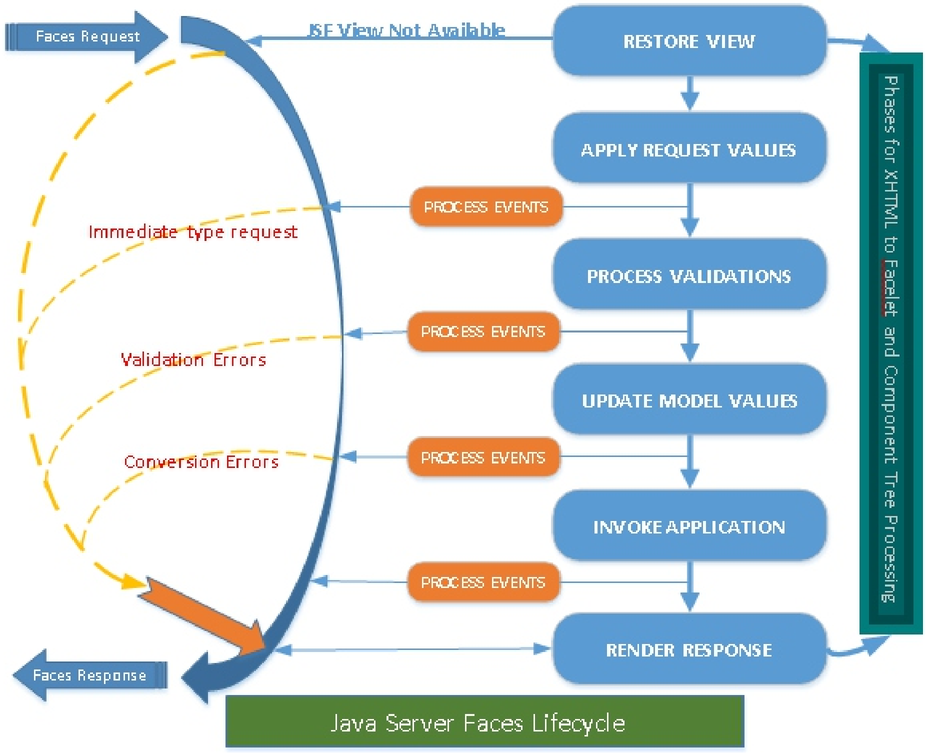


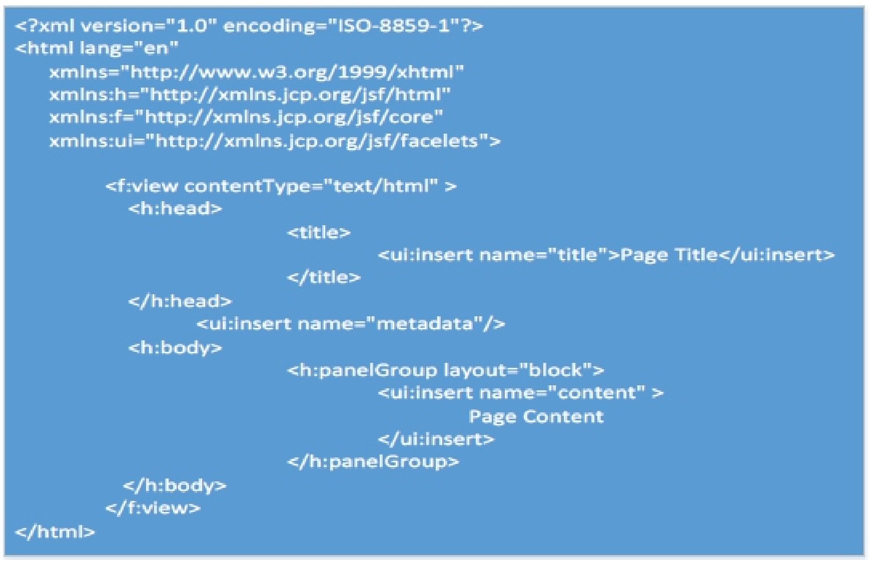
Figure 1

II. FACELETS CODE

The code outlined below are utilized to help the reader follow the processing of converting xhtml to component tree:

A. pagetemplate.xhtml

This template xhtml provides a mechanism to configure common tags for header, footer, and menu objects, for a page among others.

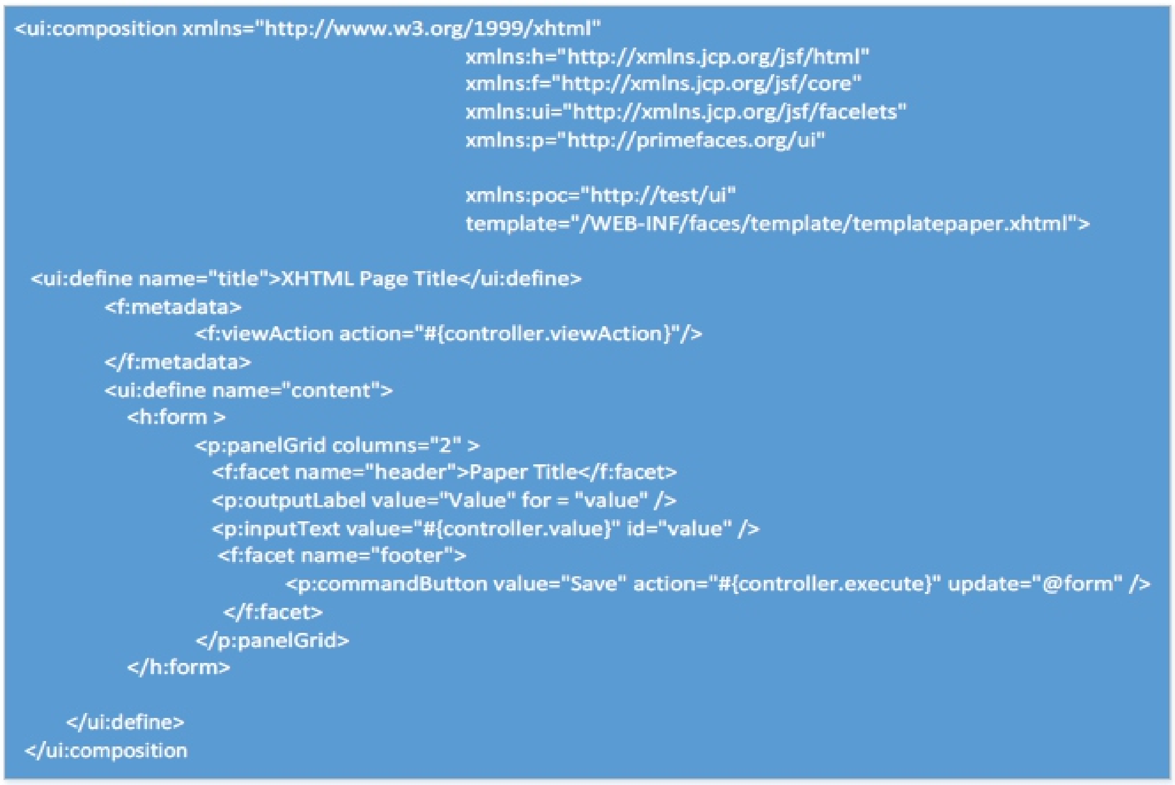


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html lang="en"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

        <f:view contentType="text/html" >
          <h:head>
                        <title>
                                <ui:insert name="title">Page Title</ui:insert>
                        </title>
          </h:head>
                <ui:insert name="metadata"/>
          <h:body>
                        <h:panelGroup layout="block">
                                <ui:insert name="content" >
                                        Page Content
                                </ui:insert>
                        </h:panelGroup>
          </h:body>
        </f:view>
</html>
```

Figure 2

B. pagetest.xhtml

This code represents the main XHTML file which will implement a certain use case, in a real-world application.

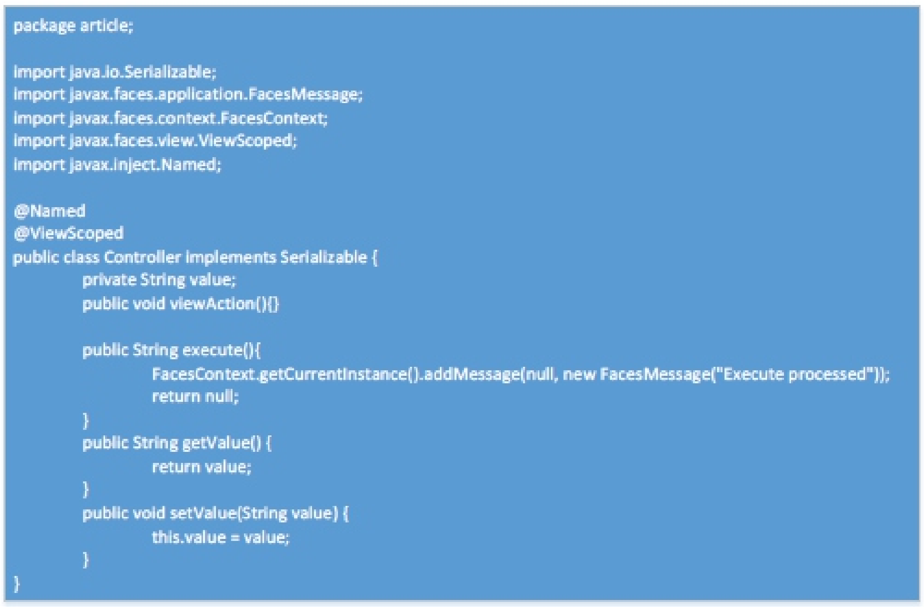


```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
                                        xmlns:h="http://xmlns.jcp.org/jsf/html"
                                        xmlns:f="http://xmlns.jcp.org/jsf/core"
                                        xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
                                        xmlns:p="http://primefaces.org/ui"

                                        xmlns:poc="http://test/ui"
                                        template="/WEB-INF/faces/template/templatepaper.xhtml">

 <ui:define name="title">XHTML Page Title</ui:define>
        <f:metadata>
                <f:viewAction action="#{controller.viewAction}"/>
        </f:metadata>
        <ui:define name="content">
          <h:form >
                <p:panelGrid columns="2" >
                  <f:facet name="header">Paper Title</f:facet>
                  <p:outputLabel value="Value" for = "value" />
                  <p:inputText value="#{controller.value}" id="value" />
                   <f:facet name="footer">
                        <p:commandButton value="Save" action="#{controller.execute}" update="@form" />
                   </f:facet>
                </p:panelGrid>
          </h:form>

     </ui:define>
 </ui:composition
```

Figure 3

C. Controller.java

Controller is a CDI annotated bean that will process the request submitted by the above defined pagetest.xhtml. The execute method is invoked during the INVOKE_APPLICATION phase.



```
package article;

import java.io.Serializable;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.faces.view.ViewScoped;
import javax.inject.Named;

@Named
@ViewScoped
public class Controller implements Serializable {
        private String value;
        public void viewAction(){}

        public String execute(){
                FacesContext.getCurrentInstance().addMessage(null, new FacesMessage("Execute processed"));
                return null;
        }
        public String getValue() {
                return value;
        }
        public void setValue(String value) {
                this.value = value;
        }
}
```

Figure 4

III. XHTML TO JSF COMPONENT TREE TRANSFORMATION

The lifecycle of a JSF application begins when a user makes an HTTP request for a page and ends when the server responds with the response.The request-response JSF lifecycle handles two kinds of requests: Initial Request and PostBack. An initial request occurs when a user makes a request for a page for the first time. A PostBack request occurs when a user submits the form contained on a page that was previously loaded into the browser because of executing an initial request.FacesServlet (provided by JSF implementation) manages therequest-processing lifecycle for web applications and initializes the resources requiredby JSF technology.Before a JSF application can start processing requests, the web container will initialize this servlet with required resources.The prerequisite to understand the transformation of XHTML to facelet is to understand how a request is handled by FacesServlet.

A. JSF ServletInitialization

Duringweb server start up, FacesServletwill get initialized.The init method of this servlet is used to initialize Factory objects such as FacesContextFactory and LifeCycleFactory. The request is handled by the service method of the FacesServlet. FacesContextis the main object that is created in this method which can then later be accessed through ThreadLocal mechanism from controller classes. FacesContextFactory has a getFacesContextmethod that creates (as needed) a new FacesContext object. The first argument of the method getFacesContextis of type Object; for the servlet request environment, this is the ServletContext object. FacesContext.getExternalContext() method returns ExternalContext, which is a wrapper around ServletContext, ServletRequest and ServletResponse. FacesContextobject is released from the thread in the finalclauseof the servlet's service method.

After the creation of the FacesContext, FacesServlet delegates further processing of the request, to the Lifecycle object. Lifecycle class has two main methodsexecute and render. The execute method processes the first five phases of the JSF Lifecycle and the render method processes the RENDER_RESPONSE phase of the lifecycle.

B. Lifecycle Execute

In the Lifecycle.execute() method, the initial JSF five phases get executed – obviously, it can return after any phase due to response Completed or renderResponse marked as true i.e.,faces Context.get Render Response () or faces Context.get ResponseComplete() returning true.

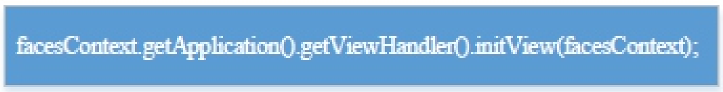
It should be noted here that if faces Context.get Response Complete() returns true then Lifecycle.render() will also not execute. For e.g., a request creates a scenario where the response will be a downloaded file, possibly in the INVOKE_APPLICATION phase, in which case the stream will be written to the outputStream. After the stream has been written,it will need to be explicitly markedfor the response as completed on the facesContextthrough facesContext.responseComplete().

The subsequent section will address the phases where a physical XHTML file is converted to Facelets and then to a component tree withUIViewRoot as the top-level component.

C. Restore View Phase (LifeCycle Execute Method – Initial Request Scenario)

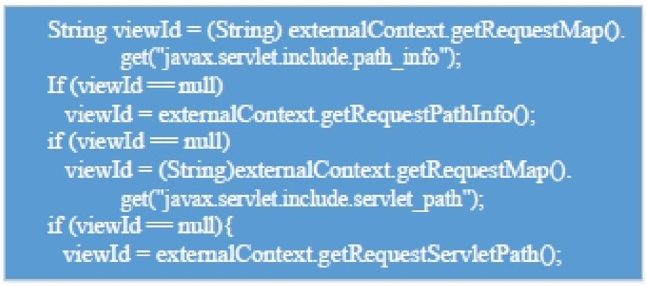
This phase plays a key role in the process of creating a Facelet butinvolves only the Facelet object corresponding to the metadata tag in the XHTML. The name of this phase suggests that it is supposed to restore the view on the PostBack request, but in case of an initial request, it creates the Facelet for the metadata tag. In the pre-phase action, the initView method on the ViewHandler is executed. This is basically to set the character encoding on the ExternalContext.

This code represents the main XHTML file which will implement a certain use case, in a real-world application.



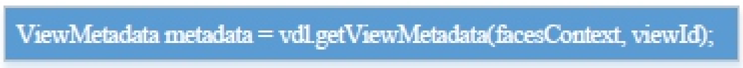
```
facesContext.getApplication().getViewHandler().initView(facesContext);
```

At this stage, the main processing in this phase of the lifecycle starts; of course, this occurs after beforePhase methods have executed for all PhaseListeners configured against this phase. Up until this point, no UIViewRoot has been created i.e.,facesContext.getViewRoot() will return null. Before this root object can get created, viewId needs to be created. In a straightforward scenario, if the initial request's URL is something like http(s)//<<server>>/<<context-root>>/article/pagetest.xhtml, then from the servlet api, the viewId for the request is determined as /article/pagetest.xhtml, which is the servlet path. The code below provides the algorithm of how a viewId is calculated for a non-portlet type of request:



```
String viewId = (String) externalContext.getRequestMap().
        get("javax.servlet.include.path_info");
If (viewId == null)
  viewId = externalContext.getRequestPathInfo();
if (viewId == null)
  viewId = (String)externalContext.getRequestMap().
        get("javax.servlet.include.servlet_path");
if (viewId == null){
  viewId = externalContext.getRequestServletPath();
```

Based on the viewId, VDL object is determined from the ViewHandler. VDL can be determined through viewHandler.getViewDeclarationLanguage(facesContext, viewId). For an XHTML based Facelet, once the VDL is determined, ViewMetaData is created via:



```
ViewMetadata metadata = vdl.getViewMetadata(facesContext, viewId);
```

The above object is a Facelet based ViewMetadatawith viewId. After the creation of the above object,createMetadataView method is executed and returns a UIViewRoot. This is the where the actual conversion from an XHTML file to a Facelet object occurs, but as stated earlier, this Facelet is only related to a metadata tag in the XHTML file.

```
UIViewRoot viewRoot = metadata.createMetadataView(facesContext)
```

The above created UIViewRoot has only UIViewAction(s) and UIViewParameter(s) as children grouped under a common parent of type facetwith name of UIViewRoot.METADATA_FACET_NAME. This facet is a direct child of UIViewRoot.Before the ViewMetadata, related components are created through its facelet, UIViewRoot needs to be created first, via

```
UIViewRoot viewRoot = facesContext.getApplication().getViewHandler().createView(facesContext, <<viewId>> );
```

```
Main processing inside createView: This method goes through the ViewDeclarationLanguage.createView(facesContext,<<viewId>>) to create the UIViewRoot.
UIViewRoot viewRoot = (UIViewRoot) facesContext.getApplication().createComponenT.(facesContext, UIViewRoot.COMPONENT_TYPE, null);
```

There is no renderer associated with the UIViewRoot component, hence the third argument is null. Based on the component type, JSFengine looks for the implementation class and creates a new instance of the component (all components, either provided by the JSF implementation or custom component) through its no-arg constructor.

```
Class<? extends UIComponent> componentClass = <<fetch the implementation class based on component type>>
UIComponent component= componentClass.newInstance();
```

During the component creation time, various annotations tagged on the component such as ListenerFor, ListenersFor, ResourceDependency and ResourceDependencies, are handled.

Once the UIViewRoot gets created, then locale and renderkitIdare set on the root object through ViewHandler class, that has methods such as calculateLocale(facesContext) andcalculate Render KitId (facesContext).

```
ViewHandler viewHandler = facesContext.getApplication.getViewHandler();
viewRoot.setLocale(viewHandler.calculateLocale(context));
viewRoot.setRenderKitId(viewHandler.calculateRenderKitId(context));
viewRoot.setViewId(<<viewId>>);
```

After the above code is executed, UIViewRootmay be used directly to access the current viewId of the request. Also, in ViewMetadata,a related Faceletis created and then based on this Facelet,UIViewRootis populated with the components related to metadata.

ViewMetadata Facelet initialization: There are two types of Facelets - the normal View Facelets and the View Metadata Facelets. TheView Metadata Facelets correspond to metadata tag in the XHTML.This part of the Faceletdoes not create the full view Facelets,because there is no need to handle other kind of tags present in the physical Facelet file (XHTML),other than the metadata.A physical Facelet file corresponds to an XHTML file, which is basically an XML file. Therefore, to get hold of a Facelet object, the XML should be first parsed. MyFaces internally uses a fast SAX compiler to achieve this parsing. The SAX compiler class is javax.xml.parsers.SAXParser and its parse method takes org.xml.sax.helpers.DefaultHandler as one of the arguments, which can handle events generated from the parser. MyFaces creates these custom handlers to handle the events generated by the parser.One of the methods from org.xml.sax.helpers.DefaultHandler to handle the transformation from an XML to Facelets, is startElement(String uri, String localName, String qName, Attributes attributes).In this method,org.xml.sax.Attributes are converted to javax.faces.view.facelets.TagAttribute. For this type of Facelet, anything other than f:metadata is not considered.Using the parameters of the method startElement,javax.faces.view.facelets.Tag object is created and passed further for processing.

TagDecorator:One of the steps of the execute method, that occurs at this point is the transformation of the Tag object using javax.faces.view.facelets.TagDecorator into a new Tag object per the decoration logic(In JSF 2.3, there is a default implementation of TagDecoratoralready available – refer to TagDecoratorin JavaDoc).The code used for this paper does not needany custom tag decoration except processing using the default TagDecoratorof JSF.

Once the XML parsing is completed, there may be various TagHandlerscomprising of Tag along with the next TagHandler. To get the Facelet from these TagHandlers, typical JSF implementations create a top level TagHandler(for e.g., EncodingHandlerin MyFaces) which becomes the starting TagHandler inside JSF implementation of Facelet. Here is the chain of TagHandlers created for this Facelet.

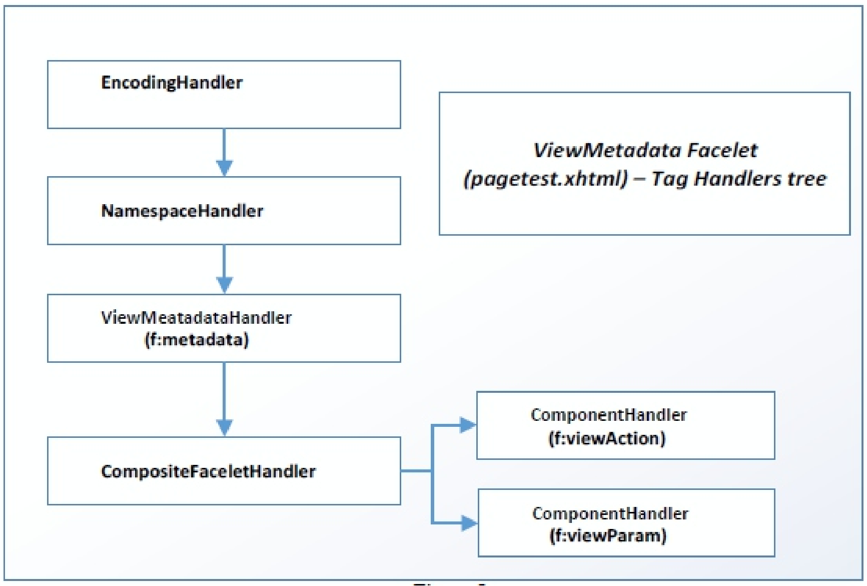

EncodingHandler
NamespaceHandler
ViewMeatadataHandler (f:metadata)
CompositeFaceletHandler
ComponentHandler (f:viewAction)
ComponentHandler (f:viewParam)
ViewMetadata Facelet (pagetest.xhtml) – Tag Handlers tree

Figure 5

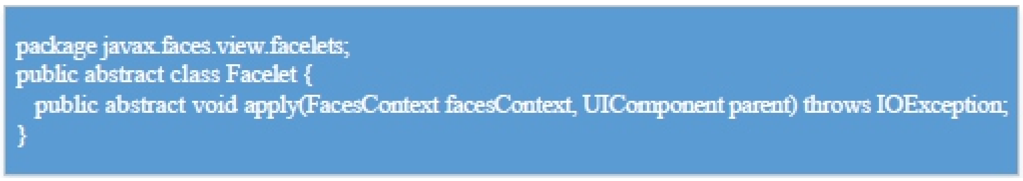
Tag Handler Factories:During the creation of the Tag Handlers, Tag Handler factories play a key role in setting up the handler objects pertaining to tag, component, converter, validator. These factories are specific to JSF implementation, but their purpose is the same, i.e., to set up or provide a mapping from XML markup on the XHTML page to its corresponding handler. For e.g.,metadatamarkup in XML page (which is associated with the namespace http://xmlns.jcp.org/jsf/core) is mapped as a TagHandler in one of the libraries (CoreLibrary), while viewAction and viewParam with the same namespace are mapped to component (without any renderer and no specific handler). When a component has no specific component handler associated with it, the JSF engine will associate ComponentHandler class as the default handler.

Like the core library, there are other types of standard libraries in each JSF implementation, such as HtmlLibrary, JstlCoreLibrary etc. which help in creating the proper mapping between XML and the namespace to its corresponding handlers. These handlers (if component handler) in turn, help in creating the actual components.

D. Render Response Phase (LifeCycle Render Method)

In response to an Initial Request (as against a PostBackrequest), this phase basically fills up the UIViewRoot with the components. However, before this occurs, the view Facelet is first created (like ViewMetadata Facelet creation in the RESTORE_VIEW phase). As described in the previous section, there are two types of Facelets - the normal View Facelets and the View Metadata Facelets. TheView Metadata Facelet is created with the help of the method buildView in VDL.The concept around building View Facelets is like building the View Metadata Faceletwith the exception that the entire XML markup is used to create tag handlers, component handlers, etc (as against using just the metadata tag in the RESTORE_VIEW Phase). Once the chain of handlers is created and set in the top level Facelet, various components get created when apply method is executed on the Facelet.

In the next figure 6, EncodingHandler is the root of the handler, which starts the building process of the component tree. The entire process starts once the apply method is executed on the Facelet(which has this EncodingHandler, as its main root handler), which in turn executes recursively the next handler and so on,until the whole chain is executed eventually creating the component tree.



```
package javax.faces.view.facelets;
public abstract class Facelet {
  public abstract void apply(FacesContext facesContext, UIComponent parent) throws IOException;
}
```

The above method in the Facelet passes the UIViewRoot as the parent component. Once a ComponentHandler is encountered, a component of that type is created and then the next handlersapply method is executed.ComponentHandler extends from DelegatingMetaTagHandlerwith the following functions, to provide the capability of calling the chained handlers in recursion till the whole tree is built.

```
public void apply(FaceletContext ctx, UIComponent parent) throws IOException{
//this method internally calls the //applyNextHandler method for chaining the next handler
            getTagHandlerDelegate().apply(ctx, parent);
   }
public void applyNextHandler (FaceletContext ctx, UIComponent c) throws IOException{
     nextHandler.apply (ctx, c);
}
```

The diagram below provides the actual tag handlers tree present in the view Facelet.

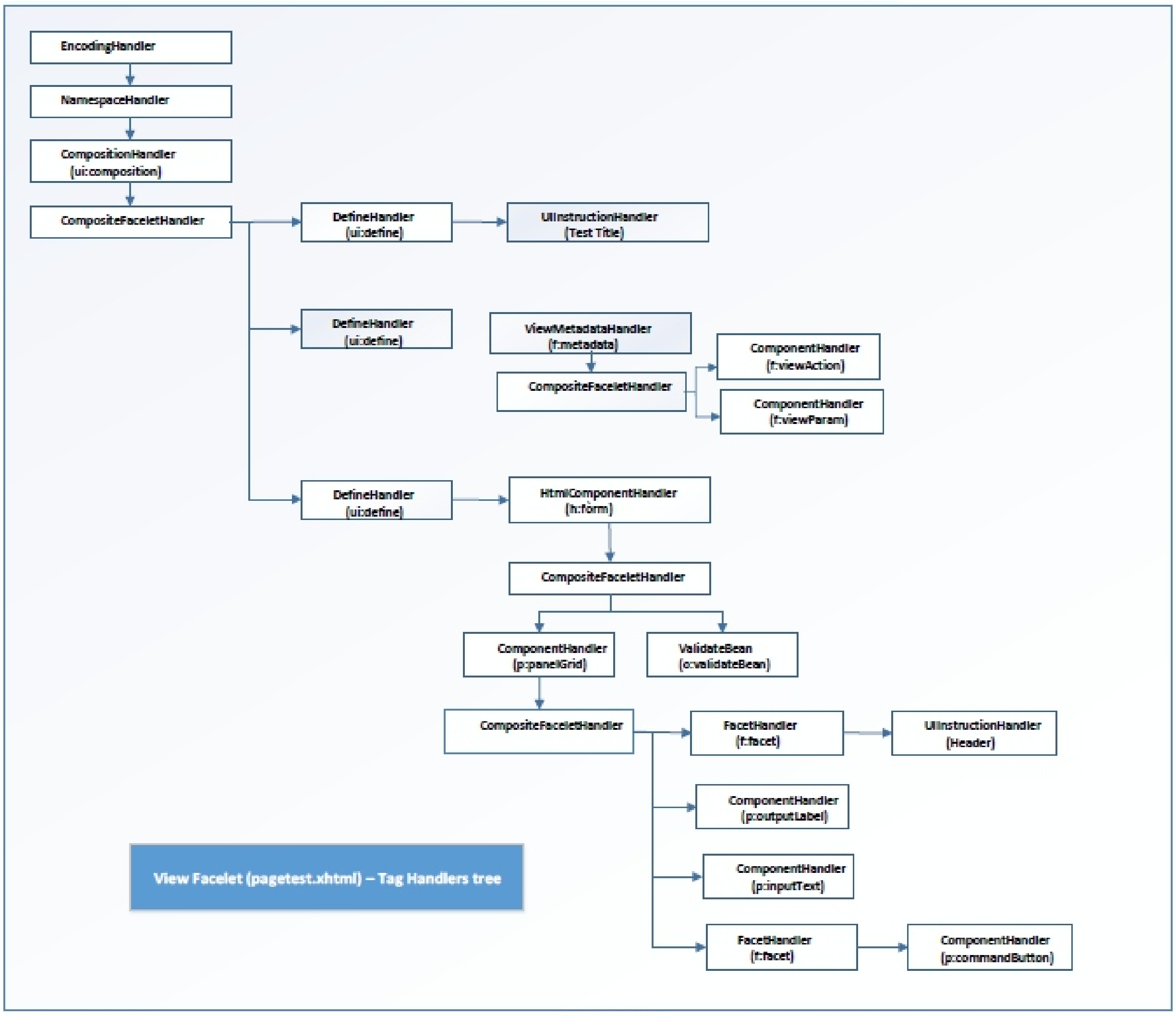

EncodingHandler
NamespaceHandler
CompositionHandler (ui:composition)
CompositeFaceletHandler
DefineHandler (ui:define)
UIInstructionHandler (Test Title)
DefineHandler (ui:define)
ViewMetadataHandler (f:metadata)
CompositeFaceletHandler
ComponentHandler (f:viewAction)
ComponentHandler (f:viewParam)
DefineHandler (ui:define)
HtmlComponentHandler (h:form)
CompositeFaceletHandler
ComponentHandler (p:panelGrid)
ValidateBean (o:validateBean)
CompositeFaceletHandler
FacetHandler (f:facet)
UIInstructionHandler (Header)
ComponentHandler (p:outputLabel)
ComponentHandler (p:inputText)
FacetHandler (f:facet)
ComponentHandler (p:commandButton)
View Facelet (pagetest.xhtml) – Tag Handlers tree

Figure 6

Since the current view has the template specified, Composition Handler includes the actual template Facelet, which can also add new handlers – this execution is explained next.

The apply method of Composition Handlerexecutes the following method to include the template Facelet. This method is present in the FaceletContext.

```
public abstract void includeFacelet(UIComponent parent, String relativePath);
```

Of course, in the above method, the UIComponent object passed will be UIViewRoot. As explained earlier in this document, the template Facelet is built using the same strategy of parsing it with the SAX compiler and building the tree.

JSF Component Tree - UIViewRoot:The diagram below depicts the component tree built in UIViewRoot, with the help of the handlers described previously:

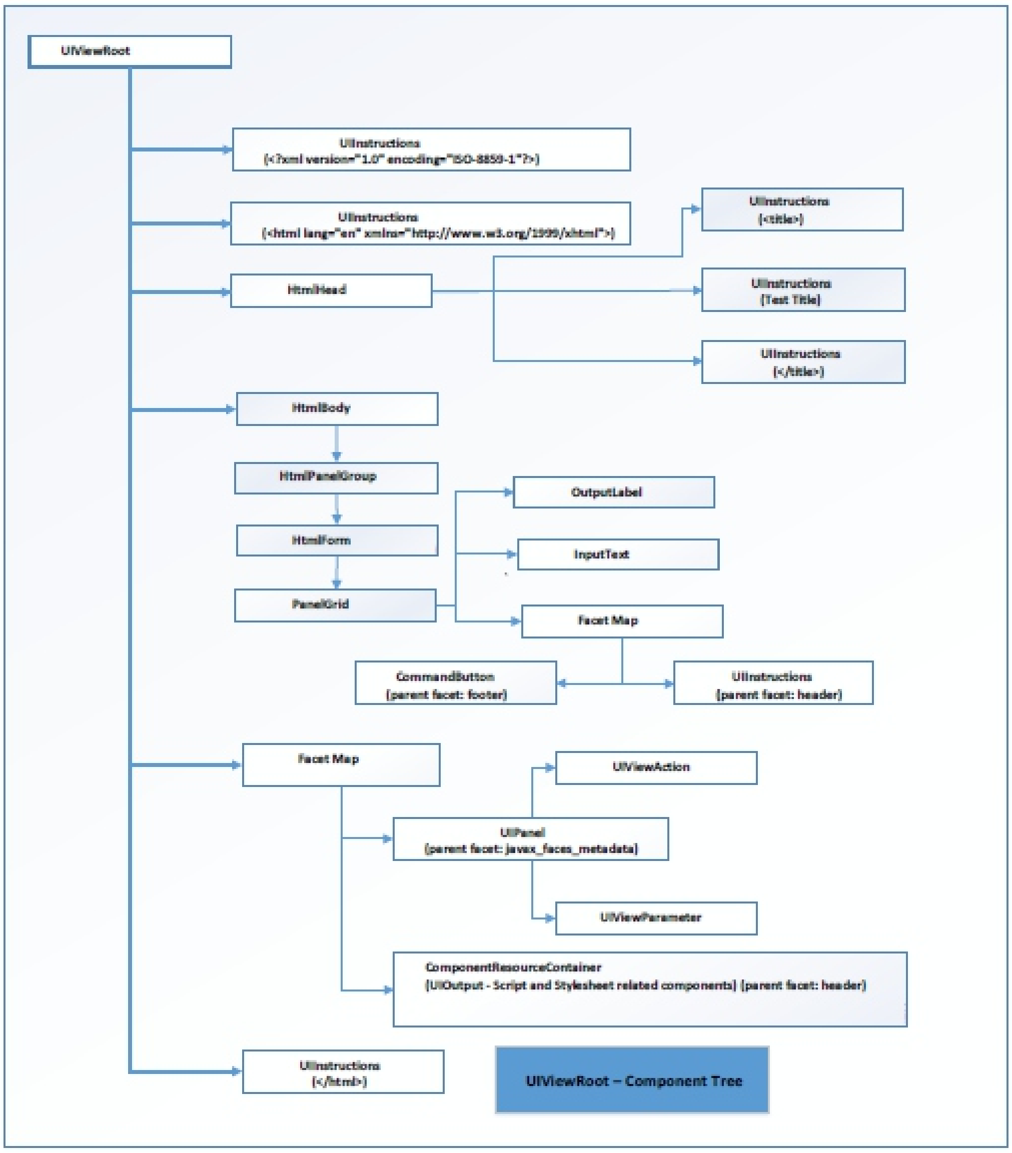

UIViewRoot
UIInstructions (<?xml version="1.0" encoding="ISO-8859-1"?>)
UIInstructions (<html lang="en" xmlns="http://www.w3.org/1999/xhtml">)
UIInstructions (<title>)
HtmlHead
UIInstructions (Test Title)
UIInstructions (</title>)
HtmlBody
HtmlPanelGroup
OutputLabel
HtmlForm
InputText
PanelGrid
Facet Map
CommandButton (parent facet: footer)
UIInstructions (parent facet: header)
Facet Map
UIViewAction
UIPanel (parent facet: javax_faces_metadata)
UIViewParameter
ComponentResourceContainer (UIOutput - Script and Stylesheet related components) (parent facet: header)
UIInstructions (</html>)
UIViewRoot – Component Tree

Figure 7

JSF Component Class Mapping:- The following is a mapping of java Classes to fully qualified Class Names (the mapping only provides classes and class names that are part of JSF API, PrimeFaces and not those that are specific to a JSF Implementation):

- UIViewRoot - javax.faces.component.UIViewRoot
- HtmlHead - javax.faces.component.html.HtmlHead
- HtmlBody - javax.faces.component.html.HtmlBody
- HtmlPanelGroup - javax.faces.component.html.HtmlPanelGroup
- HtmlForm - javax.faces.component.html.HtmlForm
- PanelGrid – org.primefaces.component.panelgrid.PanelGrid
- OuputLabel - org.primefaces.component.outputlabel.OutputLabel
- InputText - org.primefaces.component.inputtext.InputText
- CommandButton - org.primefaces.component.commandbutton.CommandButton
- UIPanel – javax.faces.component.UIPanel
- UIViewAction - javax.faces.component.UIViewAction
- UIViewParameter - javax.faces.component.UIViewParameter
- UIOutput - javax.faces.component.UIOutput
- ComponentHandler – javax.faces.view.facelets.ComponentHandler
- ResourceDependency – javax.faces.application.ResourceDependency

IV. CONCLUSION

This paper presents a thorough analysis of internals of JSF engine and covered topics such as FacesServlet initialization and how Faces Context is initialized, LifeCycle, ViewMetadata, VDL, TagDecoratorsto help readers understand how a JSF 2.3runtime eginetrans forms a physical XHTML file to multiple Facelet java objects and and then to JSF component tree as UIViewRoot. This paper provides sample code to make it easy for the readers to easily follow this complex transformation of XHTMl file. The thorough understanding of the internals of JSF engine will help software architects and designers to design and maintain complex enterprise systems based on JSF 2.3.

REFERENCES

[1] JavaServer Faces 2.3 API, website - https://javaserverfaces.github.io/docs/2.3/javadocs/index.html
[2] JavaServer Faces 2.3 Tutorial, website - https://javaee.github.io/tutorial/jsf-intro.html#BNAPH
[3] ZEEF JSF, website - https://jsf.zeef.com/arjan.tijms
[4] ZEEF JEE 8, website - https://javaee8.zeef.com/arjan.tijms
[5] PrimeFaces 7API, website - https://www.primefaces.org/docs/api/7.0/
[6] OmniFaces3.3 API, website - http://omnifaces.org/docs/javadoc/3.3/
[7] MyFaces 2.3, website - https://myfaces.apache.org/core23/index.html/

Hibernate Xdoclet Mapping Transformation to JPA

Vijay Kumar Pandey*
*Director of Technology Solutions, Intueor Consulting,
Inc. Irvine CA – USA, pandey@intueor.com

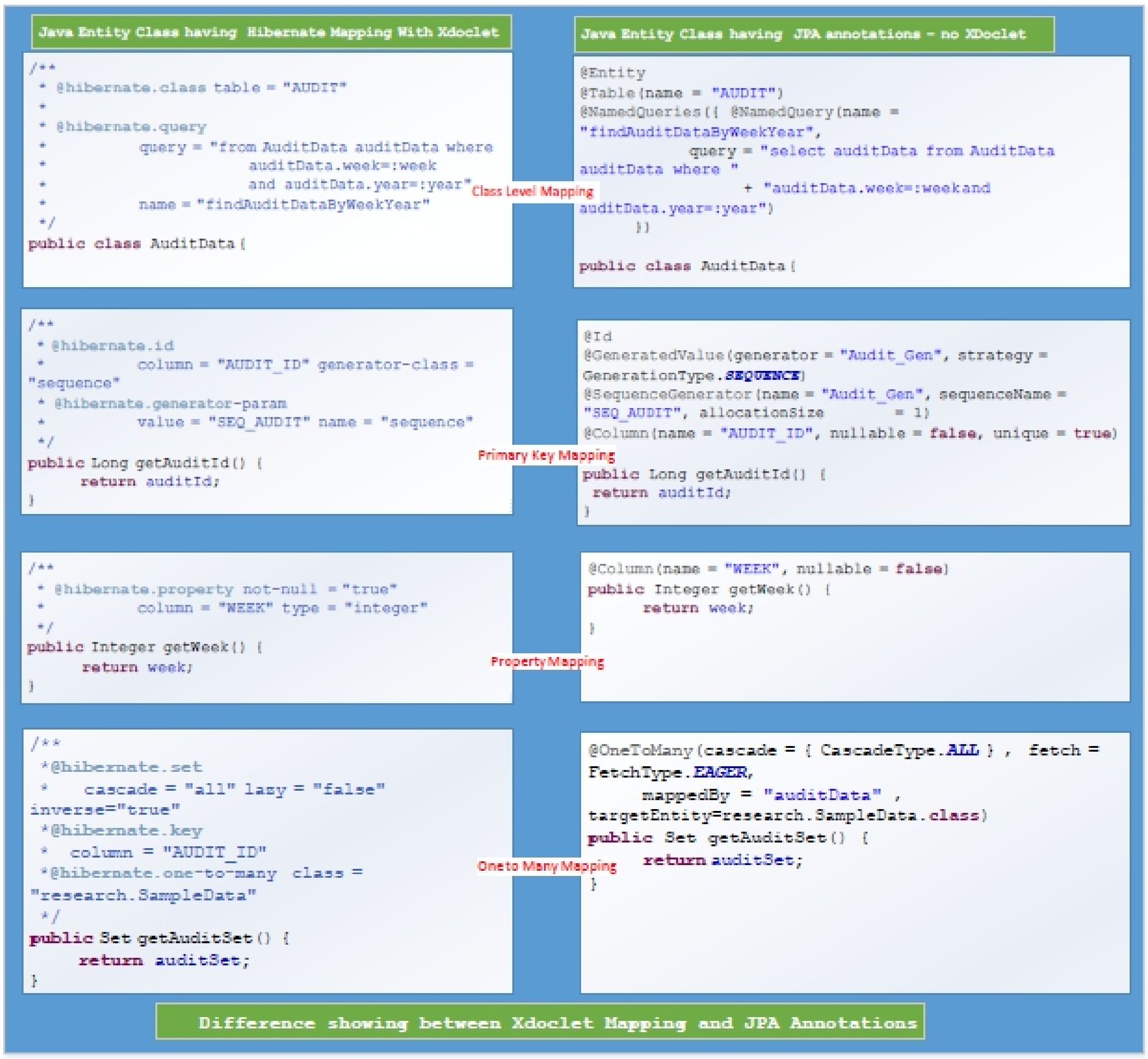
ABSTRACT

Hibernateis an open source object relational mapping (ORM) framework for the java-based technologies. Hibernate was started in 2001 before Java language annotations (feature of Java 5) came into existence, this led to a lot of enterprise systems build using Xdoclet, whichprovided a way to specify metadata as comments for Hibernate ORM within the java entity classes. Ant (a build tool) was then used to convert these Xdoclet metadata to generate hibernate mapping files (hbm files) which were then processed by the Hibernate runtime engine. With the advent of Java annotations, Hibernate also created anannotations module to provide a way to use standard java annotations within the entity classes, making the Hibernate Xdoclet annotations obsolete. Based on the community feedback and popularity of ORM based engines, Java Persistence Architecture (JPA) was first released in 2006 to provide a standard way of utilizing ORM products and its annotations. Enterprise systems that might have thousands of entities which still utilize Xdoclet way of managing metadata, have a hard time of moving to JPA as manual transformation can be buggy and expensive and that's where an automated mechanism of transforming java source codebase to not only convert Xdoclet metadata mapping to standard JPA annotations but also move the proprietary use of Hibernate API's to JPA API's across the codebase.If Hibernate engine is used as the JPA provider, one can use Hibernate API where JPA doesn't provide that specific feature.

Keywords: -ORM, JPA, Xdoclet, Hibernate, Java, API, JDBC (Java Database Connectivity)

I. INTRODUCTION

This paperprovides a detailed mechanism of how to automatically convert an Xdoclet based hibernate mapping to JPA 2.1 based annotations. One of the most important aspect of this transformation is to understand the major differences between Hibernate mapping based on Xdoclet and JPA annotations.The pictorial view below provides a side by side view of the differences between Xdoclet mapping (left side) and the JPA annotations (right side) covering areas such as mapping at the entity class, primary key property, simple property and a complex OneToMany type property.


Java Entity Class having Hibernate Mapping With Xdoclet
Java Entity Class having JPA annotations - no XDoclet
Class Level Mapping
Primary Key Mapping
Property Mapping
One to Many Mapping
Difference showing between Xdoclet Mapping and JPA Annotations

```
/**
 * @hibernate.class table = "AUDIT"
 *
 * @hibernate.query
 *           query = "from AuditData auditData where
 *                       auditData.week=:week
 *                       and auditData.year=:year"
 *           name = "findAuditDataByWeekYear"
 */
public class AuditData {
```

```
@Entity
@Table(name = "AUDIT")
@NamedQueries({ @NamedQuery(name =
"findAuditDataByWeekYear",
           query = "select auditData from AuditData
auditData where "
                 + "auditData.week=:weekand
auditData.year=:year")
      })

public class AuditData {
```

```
/**
 * @hibernate.id
 *           column = "AUDIT_ID" generator-class =
"sequence"
 * @hibernate.generator-param
 *           value = "SEQ_AUDIT" name = "sequence"
 */
public Long getAuditId() {
      return auditId;
}
```

```
@Id
@GeneratedValue(generator = "Audit_Gen", strategy =
GenerationType.SEQUENCE)
@SequenceGenerator(name = "Audit_Gen", sequenceName =
"SEQ_AUDIT", allocationSize        = 1)
@Column(name = "AUDIT_ID", nullable = false, unique = true)

public Long getAuditId() {
 return auditId;
}
```

```
/**
 * @hibernate.property not-null = "true"
 *           column = "WEEK" type = "integer"
 */
public Integer getWeek() {
      return week;
}
```

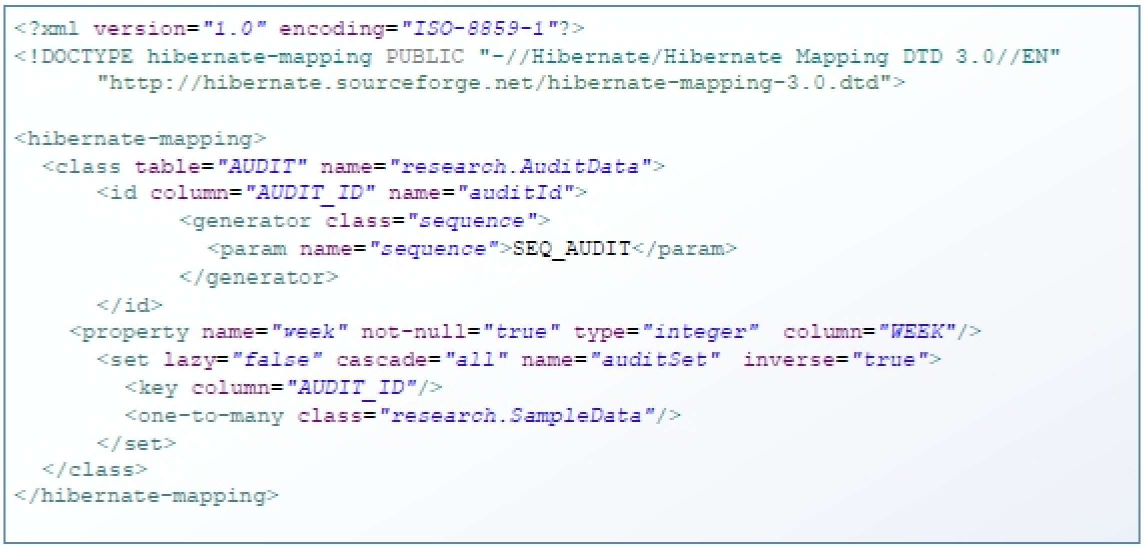
```
@Column(name = "WEEK", nullable = false)
public Integer getWeek() {
      return week;
}
```

```
/**
 *@hibernate.set
 *    cascade = "all" lazy = "false"
inverse="true"
 *@hibernate.key
 *  column = "AUDIT_ID"
 *@hibernate.one-to-many  class =
"research.SampleData"
 */
public Set getAuditSet() {
     return auditSet;
}
```

```
@OneToMany(cascade = { CascadeType.ALL } ,  fetch =
FetchType.EAGER,
     mappedBy = "auditData" ,
targetEntity=research.SampleData.class)
public Set getAuditSet() {
     return auditSet;
}
```

Figure1

The entity mapping code below shows the mapping configuration generated through Ant task for the Xdoclet mapping as shown above. The Xdoclet mapping tags cannot be read during runtime by Hibernate engine, but that's where generated mapping files provide the mapping configuration for the entity. On the other hand, any JPA provider including Hibernate can read the JPA annotations, hence no extra entity mapping file is needed.



```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class table="AUDIT" name="research.AuditData">
      <id column="AUDIT_ID" name="auditId">
            <generator class="sequence">
              <param name="sequence">SEQ_AUDIT</param>
            </generator>
      </id>
    <property name="week" not-null="true" type="integer"  column="WEEK"/>
      <set lazy="false" cascade="all" name="auditSet"  inverse="true">
        <key column="AUDIT_ID"/>
        <one-to-many class="research.SampleData"/>
      </set>
  </class>
</hibernate-mapping>
```

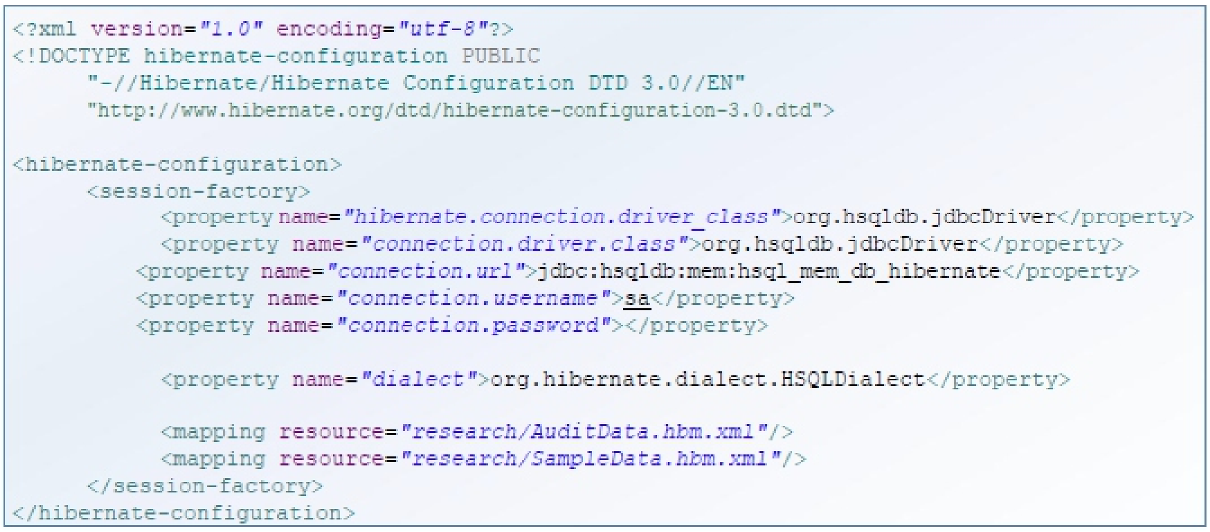
Figure 2

II. HIBERNATE INITIALIZATION - METADATA EXTRACTION

To automate the transformation of Hibernate based metadata generated through Xdoclet mapping, hibernate entity metadata needs to be extracted from the hibernate engine. This entity metadata can then be used to generate JPA based annotations. The code outlined below shows how to perform this task:

A. Hibernate Configuration File

The hibernate configuration file as shown below will be used for Hibernate to initialize its runtime.

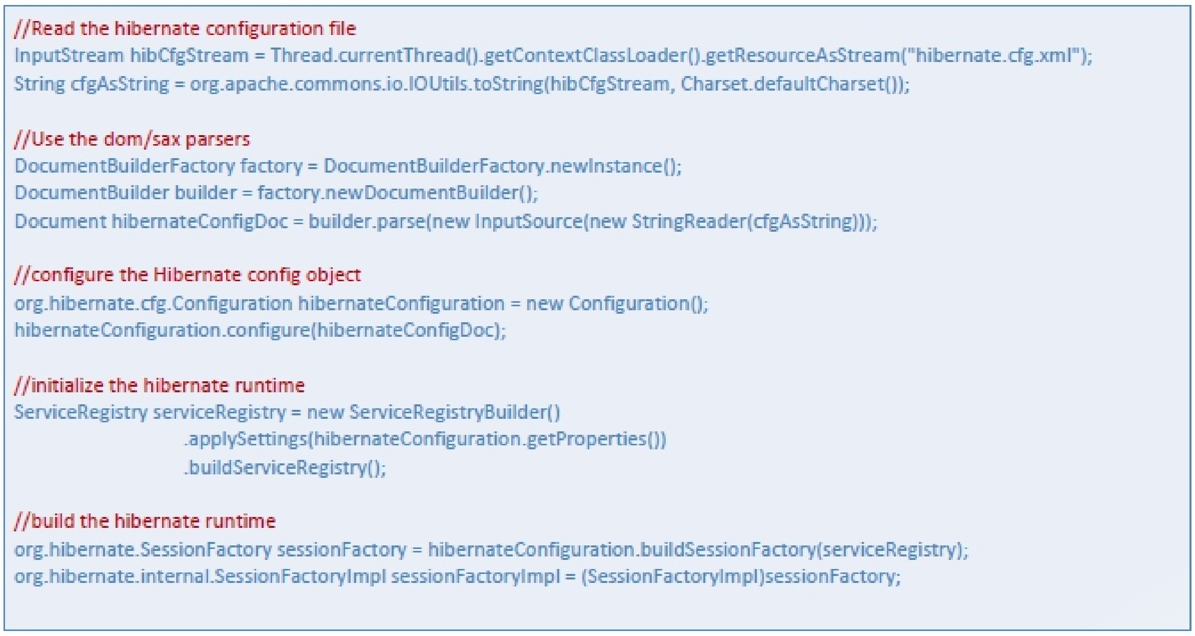


```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
      "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
      <session-factory>
            <property name="hibernate.connection.driver_class">org.hsqldb.jdbcDriver</property>
            <property name="connection.driver.class">org.hsqldb.jdbcDriver</property>
          <property name="connection.url">jdbc:hsqldb:mem:hsql_mem_db_hibernate</property>
          <property name="connection.username">sa</property>
          <property name="connection.password"></property>

            <property name="dialect">org.hibernate.dialect.HSQLDialect</property>

            <mapping resource="research/AuditData.hbm.xml"/>
            <mapping resource="research/SampleData.hbm.xml"/>
      </session-factory>
</hibernate-configuration>
```

Figure 3

B. Hibernate Runtime Initialization – Standalone Java Environment

The code below represents the Hibernate 4 way of initializing its runtime using the above file named as 'hibernate.cfg.xml'. This initialization of hibernate runtime will give access to some of the important hibernate runtime classes to read entity metadata such as org.hibernate.cfg.Configuration, org.hibernate.SessionFactory and org.hibernate.internal.SessionFactoryImpl



```
//Read the hibernate configuration file
InputStream hibCfgStream = Thread.currentThread().getContextClassLoader().getResourceAsStream("hibernate.cfg.xml");
String cfgAsString = org.apache.commons.io.IOUtils.toString(hibCfgStream, Charset.defaultCharset());

//Use the dom/sax parsers
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document hibernateConfigDoc = builder.parse(new InputSource(new StringReader(cfgAsString)));

//configure the Hibernate config object
org.hibernate.cfg.Configuration hibernateConfiguration = new Configuration();
hibernateConfiguration.configure(hibernateConfigDoc);

//initialize the hibernate runtime
ServiceRegistry serviceRegistry = new ServiceRegistryBuilder()
                        .applySettings(hibernateConfiguration.getProperties())
                        .buildServiceRegistry();

//build the hibernate runtime
org.hibernate.SessionFactory sessionFactory = hibernateConfiguration.buildSessionFactory(serviceRegistry);
org.hibernate.internal.SessionFactoryImpl sessionFactoryImpl = (SessionFactoryImpl)sessionFactory;
```

Figure 4

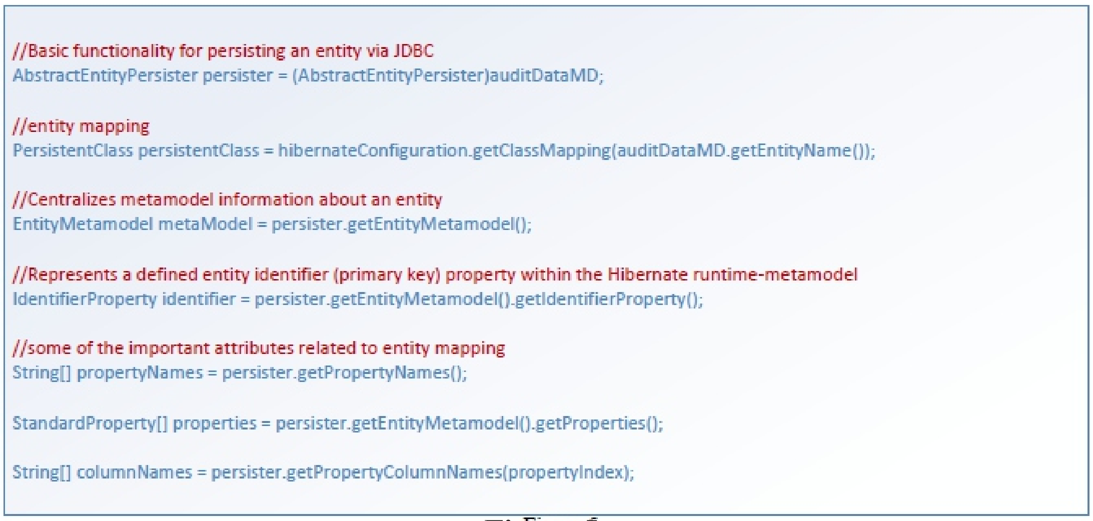
C. Hibernate Entity Metadata

Hibernate keeps the entity metadata as part of its class 'org.hibernate.metadata.ClassMetadata'. The SessionFactory built in the above step provides a way to access the entity metadata.

```
Map<String, ClassMetadata> metadatas =sessionFactory.getAllClassMetadata();
ClassMetadata auditDataMD = metadatas.get("research.AuditData");
```

D. Parts of Entity Metadata

Once the entity metadata is accessed for a specific entity, it can be used to gain access to metadata for its various properties, identifier (primary key) property. The code below shows the mechanism of how to access these property level metadata.

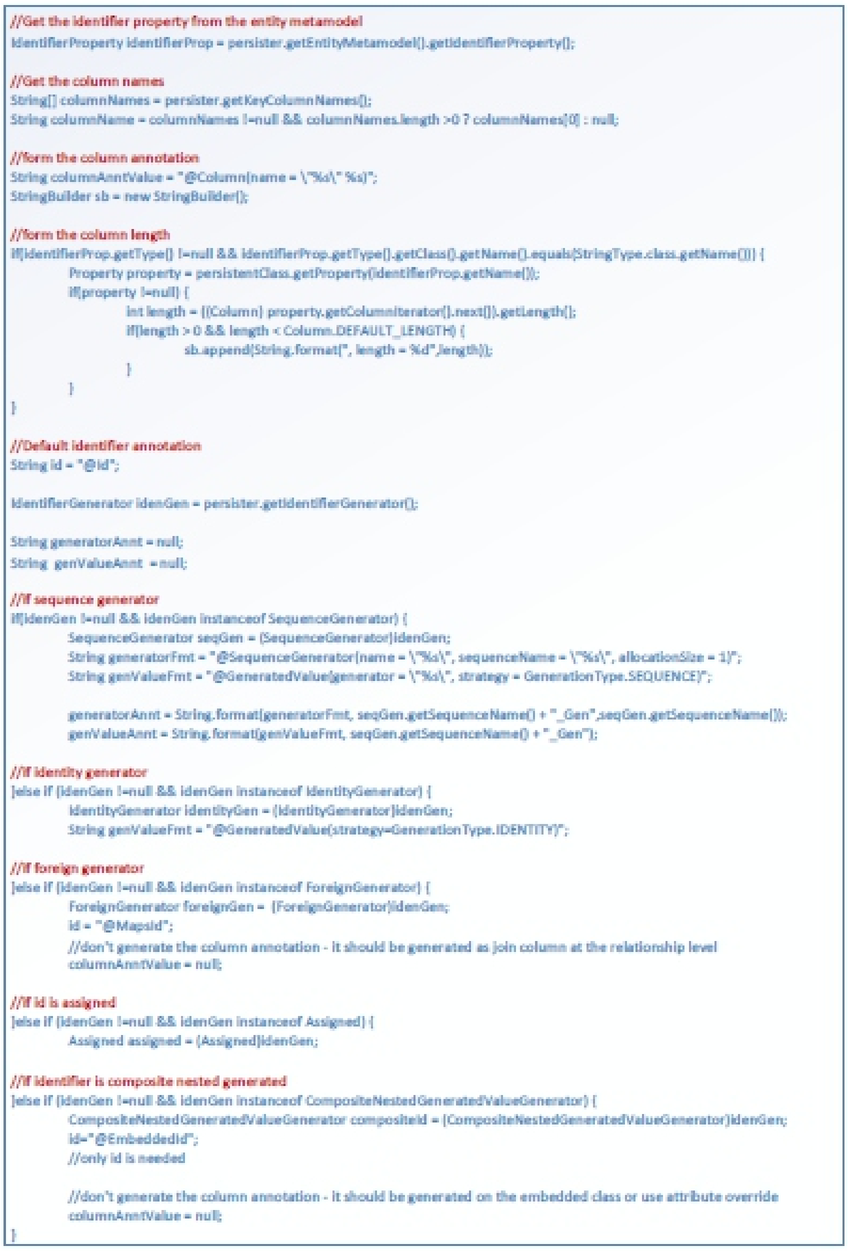


```
//Basic functionality for persisting an entity via JDBC
AbstractEntityPersister persister = (AbstractEntityPersister)auditDataMD;

//entity mapping
PersistentClass persistentClass = hibernateConfiguration.getClassMapping(auditDataMD.getEntityName());

//Centralizes metamodel information about an entity
EntityMetamodel metaModel = persister.getEntityMetamodel();

//Represents a defined entity identifier (primary key) property within the Hibernate runtime-metamodel
IdentifierProperty identifier = persister.getEntityMetamodel().getIdentifierProperty();

//some of the important attributes related to entity mapping
String[] propertyNames = persister.getPropertyNames();

StandardProperty[] properties = persister.getEntityMetamodel().getProperties();

String[] columnNames = persister.getPropertyColumnNames(propertyIndex);
```

Figure 5

E. Generate JPA Identifier Annotations from Metadata

Once identifier property is accessed as shown above, its related metadata can be extracted and with that information JPA annotations can be formed. The algorithmwith pseudo code below shows this process.

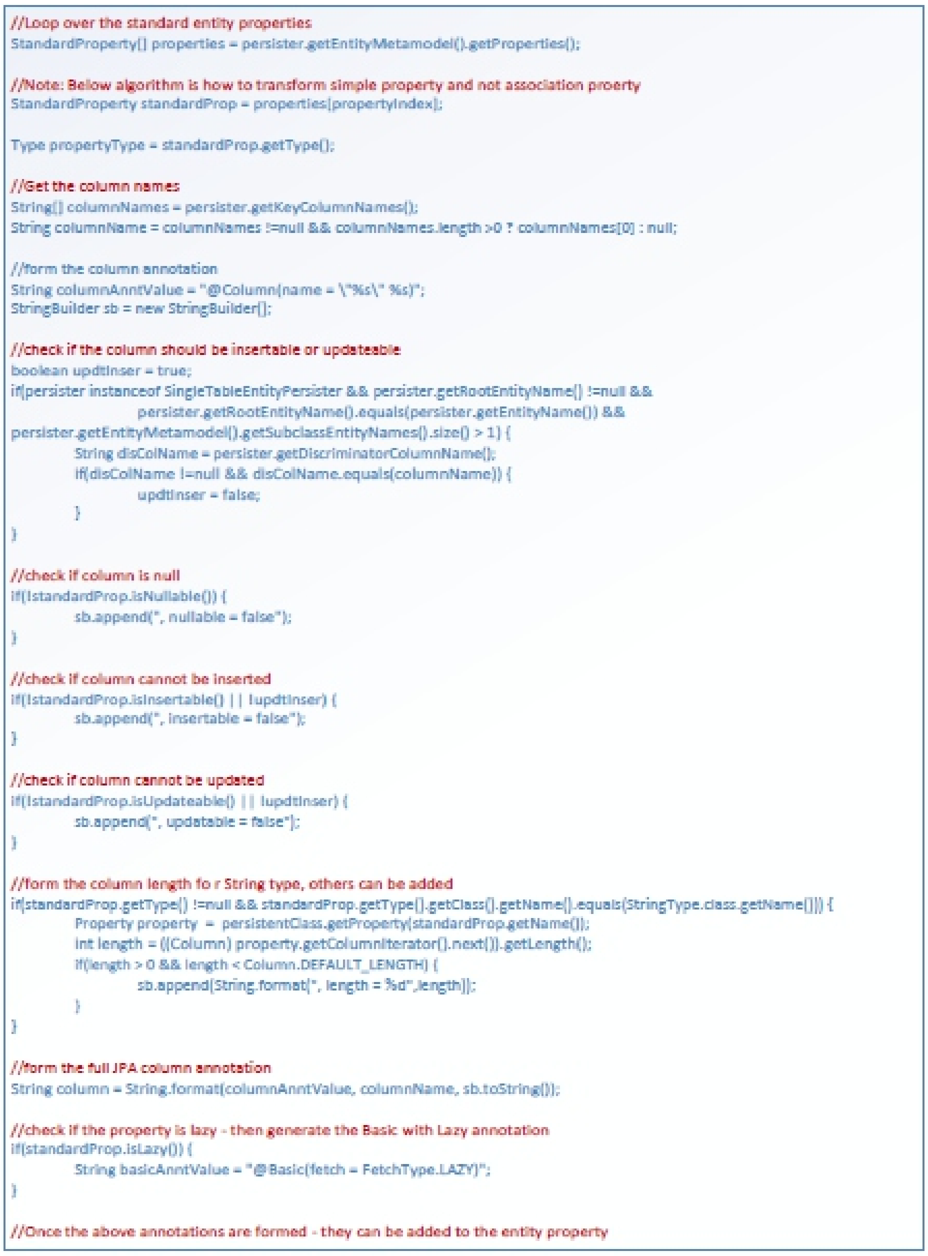


```
//Get the identifier property from the entity metamodel
IdentifierProperty identifierProp = persister.getEntityMetamodel().getIdentifierProperty();

//Get the column names
String[] columnNames = persister.getKeyColumnNames();
String columnName = columnNames !=null && columnNames.length >0 ? columnNames[0] : null;

//form the column annotation
String columnAnntValue = "@Column(name = \"%s\" %s)";
StringBuilder sb = new StringBuilder();

//form the column length
if(identifierProp.getType() !=null && identifierProp.getType().getClass().getName().equals(StringType.class.getName())) {
          Property property = persistentClass.getProperty(identifierProp.getName());
          if(property !=null) {
                    int length = ((Column) property.getColumnIterator().next()).getLength();
                    if(length > 0 && length < Column.DEFAULT_LENGTH) {
                              sb.append(String.format(", length = %d",length));
                    }
          }
}

//Default identifier annotation
String id = "@Id";

IdentifierGenerator idenGen = persister.getIdentifierGenerator();

String generatorAnnt = null;
String  genValueAnnt  =null;

//If sequence generator
if(idenGen !=null && idenGen instanceof SequenceGenerator) {
          SequenceGenerator seqGen = (SequenceGenerator)idenGen;
          String generatorFmt = "@SequenceGenerator(name = \"%s\", sequenceName = \"%s\", allocationSize = 1)";
          String genValueFmt = "@GeneratedValue(generator = \"%s\", strategy = GenerationType.SEQUENCE)";

          generatorAnnt = String.format(generatorFmt, seqGen.getSequenceName() + "_Gen",seqGen.getSequenceName());
          genValueAnnt = String.format(genValueFmt, seqGen.getSequenceName() + "_Gen");

//If identity generator
}else if (idenGen !=null && idenGen instanceof IdentityGenerator) {
          IdentityGenerator identityGen = (IdentityGenerator)idenGen;
          String genValueFmt = "@GeneratedValue(strategy=GenerationType.IDENTITY)";

//If foreign generator
}else if (idenGen !=null && idenGen instanceof ForeignGenerator) {
          ForeignGenerator foreignGen =  (ForeignGenerator)idenGen;
          id = "@MapsId";
          //don't generate the column annotation - it should be generated as join column at the relationship level
          columnAnntValue = null;

//If id is assigned
}else if (idenGen !=null && idenGen instanceof Assigned) {
          Assigned assigned = (Assigned)idenGen;

//If identifier is composite nested generated
}else if (idenGen !=null && idenGen instanceof CompositeNestedGeneratedValueGenerator) {
          CompositeNestedGeneratedValueGenerator compositeId = (CompositeNestedGeneratedValueGenerator)idenGen;
          id="@EmbeddedId";
          //only id is needed

          //don't generate the column annotation - it should be generated on the embedded class or use attribute override
          columnAnntValue = null;
}
```

Figure 6

F. Generate JPA Non-IdentifierProperty Annotations from Metadata

This section details how to generate JPA annotations of non-association standard property of an entity with the help of its metadata. Once the JPA annotations are formed, it could easily be added to the JPA based entities. The algorithm with the pseudo code describes how to achieve the formation of the annotations.

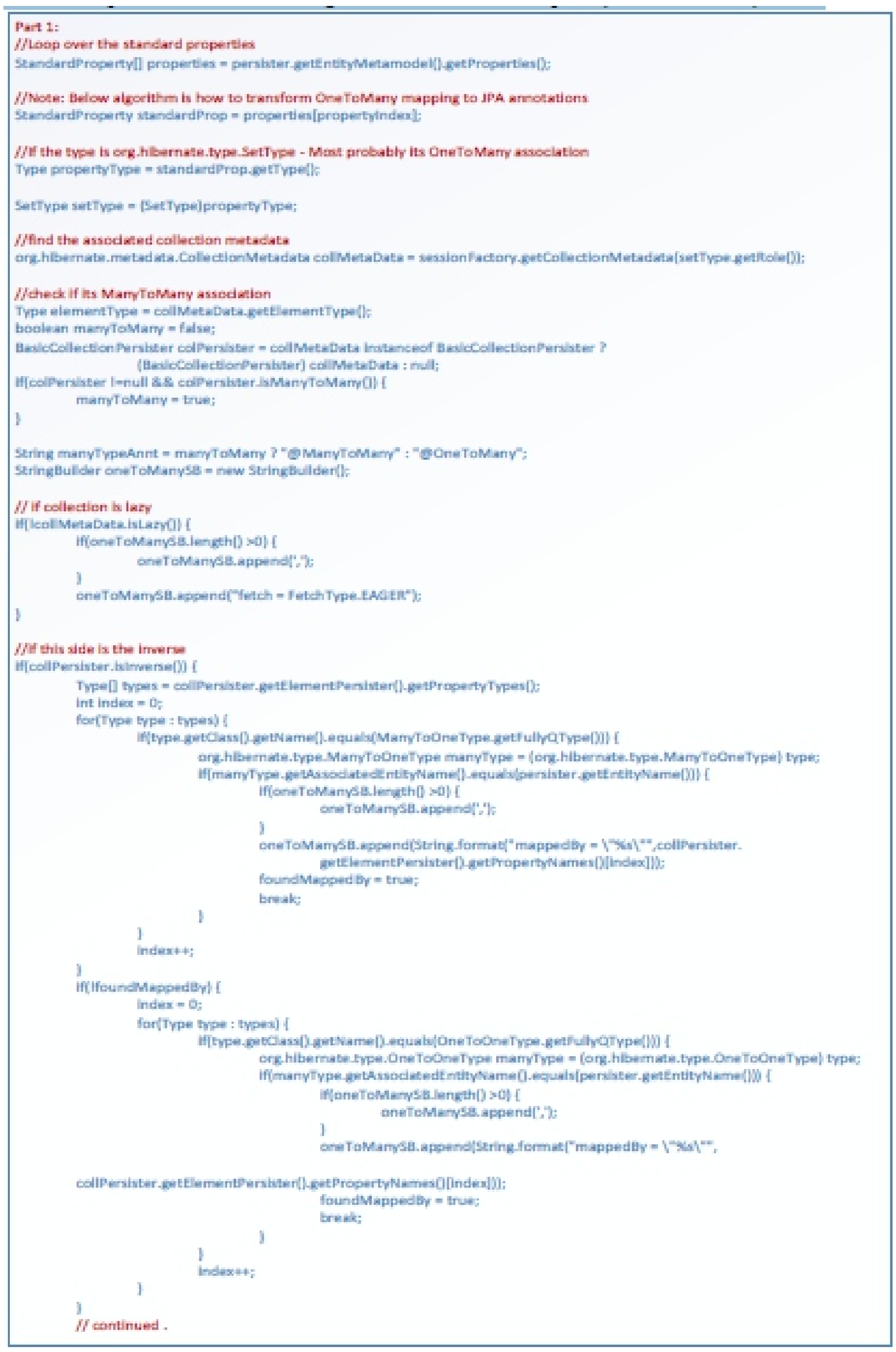


```
//Loop over the standard entity properties
StandardProperty[] properties = persister.getEntityMetamodel().getProperties();

//Note: Below algorithm is how to transform simple property and not association proerty
StandardProperty standardProp = properties[propertyIndex];

Type propertyType = standardProp.getType();

//Get the column names
String[] columnNames = persister.getKeyColumnNames();
String columnName = columnNames !=null && columnNames.length >0 ? columnNames[0] : null;

//form the column annotation
String columnAnntValue = "@Column(name = \"%s\" %s)";
StringBuilder sb = new StringBuilder();

//check if the column should be insertable or updateable
boolean updtInser = true;
if(persister instanceof SingleTableEntityPersister && persister.getRootEntityName() !=null &&
                        persister.getRootEntityName().equals(persister.getEntityName()) &&
persister.getEntityMetamodel().getSubclassEntityNames().size() > 1) {
            String disColName = persister.getDiscriminatorColumnName();
            if(disColName !=null && disColName.equals(columnName)) {
                        updtInser = false;
            }
}

//check if column is null
if(!standardProp.isNullable()) {
            sb.append(", nullable = false");
}

//check if column cannot be inserted
if(!standardProp.isInsertable() || !updtInser) {
            sb.append(", insertable = false");
}

//check if column cannot be updated
if(!standardProp.isUpdateable() || !updtInser) {
            sb.append(", updatable = false");
}

//form the column length fo r String type, others can be added
if(standardProp.getType() !=null && standardProp.getType().getClass().getName().equals(StringType.class.getName())) {
            Property property  =  persistentClass.getProperty(standardProp.getName());
            int length = ((Column) property.getColumnIterator().next()).getLength();
            if(length > 0 && length < Column.DEFAULT_LENGTH) {
                        sb.append(String.format(", length = %d",length));
            }
}

//form the full JPA column annotation
String column = String.format(columnAnntValue, columnName, sb.toString());

//check if the property is lazy - then generate the Basic with Lazy annotation
if(standardProp.isLazy()) {
            String basicAnntValue = "@Basic(fetch = FetchType.LAZY)";
}

//Once the above annotations are formed - they can be added to the entity property
```

Figure 7

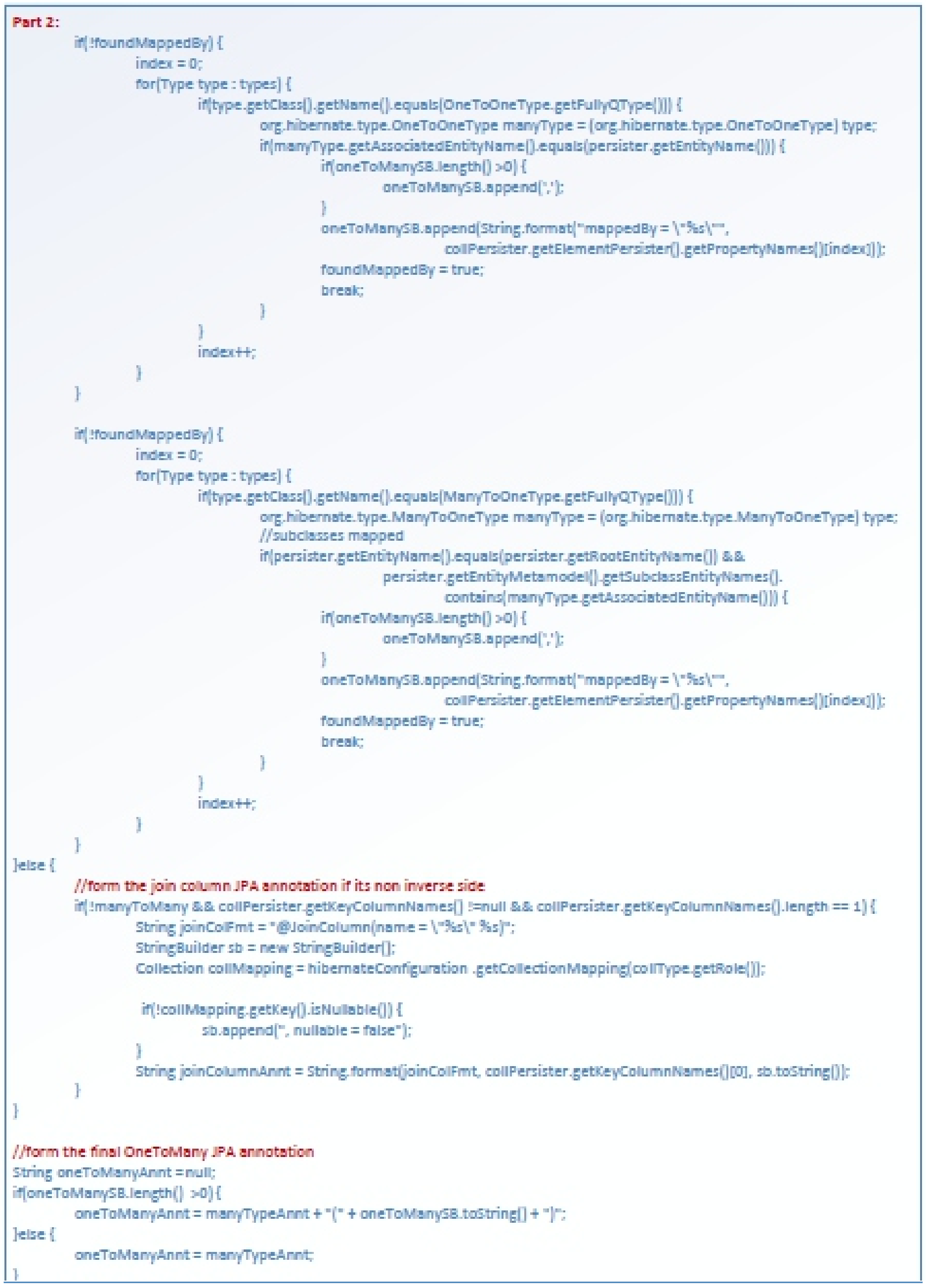
G. Generate JPA association OneToMany Property Annotations from Metadata

This section details how to generate JPA annotations of association property such as OneToManyof an entity with the help of its metadata. This algorithm is divided in two parts (Part 1 and Part 2) below.



```
Part 1:
//Loop over the standard properties
StandardProperty[] properties = persister.getEntityMetamodel().getProperties();

//Note: Below algorithm is how to transform OneToMany mapping to JPA annotations
StandardProperty standardProp = properties[propertyIndex];

//If the type is org.hibernate.type.SetType - Most probably its OneToMany association
Type propertyType = standardProp.getType();

SetType setType = (SetType)propertyType;

//find the associated collection metadata
org.hibernate.metadata.CollectionMetadata collMetaData = sessionFactory.getCollectionMetadata(setType.getRole());

//check if its ManyToMany association
Type elementType = collMetaData.getElementType();
boolean manyToMany = false;
BasicCollectionPersister colPersister = collMetaData instanceof BasicCollectionPersister ?
                (BasicCollectionPersister) collMetaData : null;
if(colPersister !=null && colPersister.isManyToMany()) {
        manyToMany = true;
}

String manyTypeAnnt = manyToMany ? "@ManyToMany" : "@OneToMany";
StringBuilder oneToManySB = new StringBuilder();

// if collection is lazy
if(!collMetaData.isLazy()) {
        if(oneToManySB.length() >0) {
                oneToManySB.append(',');
        }
        oneToManySB.append("fetch = FetchType.EAGER");
}

//if this side is the inverse
if(collPersister.isInverse()) {
        Type[] types = collPersister.getElementPersister().getPropertyTypes();
        int index = 0;
        for(Type type : types) {
                if(type.getClass().getName().equals(ManyToOneType.getFullyQType())) {
                        org.hibernate.type.ManyToOneType manyType = (org.hibernate.type.ManyToOneType) type;
                        if(manyType.getAssociatedEntityName().equals(persister.getEntityName())) {
                                if(oneToManySB.length() >0) {
                                        oneToManySB.append(',');
                                }
                                oneToManySB.append(String.format("mappedBy = \"%s\"",collPersister.
                                        getElementPersister().getPropertyNames()[index]));
                                foundMappedBy = true;
                                break;
                        }
                }
                index++;
        }
        if(!foundMappedBy) {
                index = 0;
                for(Type type : types) {
                        if(type.getClass().getName().equals(OneToOneType.getFullyQType())) {
                                org.hibernate.type.OneToOneType manyType = (org.hibernate.type.OneToOneType) type;
                                if(manyType.getAssociatedEntityName().equals(persister.getEntityName())) {
                                        if(oneToManySB.length() >0) {
                                                oneToManySB.append(',');
                                        }
                                        oneToManySB.append(String.format("mappedBy = \"%s\"",

        collPersister.getElementPersister().getPropertyNames()[index]));
                                        foundMappedBy = true;
                                        break;
                                }
                        }
                        index++;
                }
        }
        // continued .
```

Figure 8



```
Part 2:
        if(!foundMappedBy) {
                index = 0;
                for(Type type : types) {
                        if(type.getClass().getName().equals(OneToOneType.getFullyQType())) {
                                org.hibernate.type.OneToOneType manyType = (org.hibernate.type.OneToOneType) type;
                                if(manyType.getAssociatedEntityName().equals(persister.getEntityName())) {
                                        if(oneToManySB.length() >0) {
                                                oneToManySB.append(',');
                                        }
                                        oneToManySB.append(String.format("mappedBy = \"%s\"",
                                                        collPersister.getElementPersister().getPropertyNames()[index]));
                                        foundMappedBy = true;
                                        break;
                                }
                        }
                        index++;
                }
        }

        if(!foundMappedBy) {
                index = 0;
                for(Type type : types) {
                        if(type.getClass().getName().equals(ManyToOneType.getFullyQType())) {
                                org.hibernate.type.ManyToOneType manyType = (org.hibernate.type.ManyToOneType) type;
                                //subclasses mapped
                                if(persister.getEntityName().equals(persister.getRootEntityName()) &&
                                                persister.getEntityMetamodel().getSubclassEntityNames().
                                                        contains(manyType.getAssociatedEntityName())) {
                                        if(oneToManySB.length() >0) {
                                                oneToManySB.append(',');
                                        }
                                        oneToManySB.append(String.format("mappedBy = \"%s\"",
                                                        collPersister.getElementPersister().getPropertyNames()[index]));
                                        foundMappedBy = true;
                                        break;
                                }
                        }
                        index++;
                }
        }
}else {
        //form the join column JPA annotation if its non inverse side
        if(!manyToMany && collPersister.getKeyColumnNames() !=null && collPersister.getKeyColumnNames().length == 1) {
                String joinColFmt = "@JoinColumn(name = \"%s\" %s)";
                StringBuilder sb = new StringBuilder();
                Collection collMapping = hibernateConfiguration .getCollectionMapping(collType.getRole());

                 if(!collMapping.getKey().isNullable()) {
                        sb.append(", nullable = false");
                }
                String joinColumnAnnt = String.format(joinColFmt, collPersister.getKeyColumnNames()[0], sb.toString());
        }
}

//form the final OneToMany JPA annotation
String oneToManyAnnt =null;
if(oneToManySB.length()  >0){
        oneToManyAnnt = manyTypeAnnt + "(" + oneToManySB.toString() + ")";
}else {
        oneToManyAnnt = manyTypeAnnt;
}
```

Figure 9

III. CONCLUSION

This paper presents a thorough analysis of how to extract the hibernate entity metadata that were provided through Xdoclet and then converted to hibernate mapping configuration. This paper also deep dives into the internals of the Hibernate 4 engine to extract this metadata information for generating the JPA annotations. Property types such as identifier, a standard and a collection based OneToMany association-based properties were thoroughly investigated and detailed algorithm along with pseudo code was provided to unravel the complexities of how to convert these ORM provider specific mapping to standard JPA annotations which could be easily analysed by any ORM provider. The thorough understanding of this transformation process can help enterprise systems to transform to standard JPA annotations that are still using now legacy Xdoclet along with non-standard hibernate mapping files.

REFERENCES

[1] Hibernate 4.2, website - https://hibernate.org/orm/documentation/4.2/
[2] Hibernate 4.2 API, website - http://docs.jboss.org/hibernate/orm/4.2/javadocs/
[3] JPA 2.1 API, website - http://docs.jboss.org/hibernate/jpa/2.1/api/
[4] Hibernate Xdoclet, website - http://xdoclet.sourceforge.net/xdoclet/tags/hibernate-tags.html
[5] JPA 2.1 Specification, website - https://download.oracle.com/otndocs/jcp/persistence-2_1-fr-eval-spec/index.html
[6] JPA Annotations, website - https://www.objectdb.com/api/java/jpa/annotations

Instructions for Authors

Essentials for Publishing in this Journal

1 Submitted articles should not have been previously published or be currently under consideration for publication elsewhere.

2 Conference papers may only be submitted if the paper has been completely re-written (taken to mean more than 50%) and the author has cleared any necessary permission with the copyright owner if it has been previously copyrighted.

3 All our articles are refereed through a double-blind process.

4 All authors must declare they have read and agreed to the content of the submitted article and must sign a declaration correspond to the originality of the article.

Submission Process

All articles for this journal must be submitted using our online submissions system. http://enrichedpub.com/ . Please use the Submit Your Article link in the Author Service area.

Manuscript Guidelines

The instructions to authors about the article preparation for publication in the Manuscripts are submitted online, through the e-Ur (Electronic editing) system, developed by **Enriched Publications Pvt. Ltd**. The article should contain the abstract with keywords, introduction, body, conclusion, references and the summary in English language (without heading and subheading enumeration). The article length should not exceed 16 pages of A4 paper format.

Title

The title should be informative. It is in both Journal's and author's best interest to use terms suitable. For indexing and word search. If there are no such terms in the title, the author is strongly advised to add a subtitle. The title should be given in English as well. The titles precede the abstract and the summary in an appropriate language.

Letterhead Title

The letterhead title is given at a top of each page for easier identification of article copies in an Electronic form in particular. It contains the author's surname and first name initial .article title, journal title and collation (year, volume, and issue, first and last page). The journal and article titles can be given in a shortened form.

Author's Name

Full name(s) of author(s) should be used. It is advisable to give the middle initial. Names are given in their original form.

Contact Details

The postal address or the e-mail address of the author (usually of the first one if there are more Authors) is given in the footnote at the bottom of the first page.

Type of Articles

Classification of articles is a duty of the editorial staff and is of special importance. Referees and the members of the editorial staff, or section editors, can propose a category, but the editor-in-chief has the sole responsibility for their classification. Journal articles are classified as follows:

Scientific articles:

1. Original scientific paper (giving the previously unpublished results of the author's own research based on management methods).

2. Survey paper (giving an original, detailed and critical view of a research problem or an area to which the author has made a contribution visible through his self-citation);

3. Short or preliminary communication (original management paper of full format but of a smaller extent or of a preliminary character);

4. Scientific critique or forum (discussion on a particular scientific topic, based exclusively on management argumentation) and commentaries. Exceptionally, in particular areas, a scientific paper in the Journal can be in a form of a monograph or a critical edition of scientific data (historical, archival, lexicographic, bibliographic, data survey, etc.) which were unknown or hardly accessible for scientific research.

Professional articles:

1. Professional paper (contribution offering experience useful for improvement of professional practice but not necessarily based on scientific methods);
2. Informative contribution (editorial, commentary, etc.);
3. Review (of a book, software, case study, scientific event, etc.)

Language

The article should be in English. The grammar and style of the article should be of good quality. The systematized text should be without abbreviations (except standard ones). All measurements must be in SI units. The sequence of formulae is denoted in Arabic numerals in parentheses on the right-hand side.

Abstract and Summary

An abstract is a concise informative presentation of the article content for fast and accurate Evaluation of its relevance. It is both in the Editorial Office's and the author's best interest for an abstract to contain terms often used for indexing and article search. The abstract describes the purpose of the study and the methods, outlines the findings and state the conclusions. A 100- to 250-Word abstract should be placed between the title and the keywords with the body text to follow. Besides an abstract are advised to have a summary in English, at the end of the article, after the Reference list. The summary should be structured and long up to 1/10 of the article length (it is more extensive than the abstract).

Keywords

Keywords are terms or phrases showing adequately the article content for indexing and search purposes. They should be allocated heaving in mind widely accepted international sources (index, dictionary or thesaurus), such as the Web of Science keyword list for science in general. The higher their usage frequency is the better. Up to 10 keywords immediately follow the abstract and the summary, in respective languages.

Acknowledgements

The name and the number of the project or programmed within which the article was realized is given in a separate note at the bottom of the first page together with the name of the institution which financially supported the project or programmed.

Tables and Illustrations

All the captions should be in the original language as well as in English, together with the texts in illustrations if possible. Tables are typed in the same style as the text and are denoted by numerals at the top. Photographs and drawings, placed appropriately in the text, should be clear, precise and suitable for reproduction. Drawings should be created in Word or Corel.

Citation in the Text

Citation in the text must be uniform. When citing references in the text, use the reference number set in square brackets from the Reference list at the end of the article.

Footnotes

Footnotes are given at the bottom of the page with the text they refer to. They can contain less relevant details, additional explanations or used sources (e.g. scientific material, manuals). They cannot replace the cited literature.
The article should be accompanied with a cover letter with the information about the author(s): surname, middle initial, first name, and citizen personal number, rank, title, e-mail address, and affiliation address, home address including municipality, phone number in the office and at home (or a mobile phone number). The cover letter should state the type of the article and tell which illustrations are original and which are not.

Address of the Editorial Office:

Enriched Publications Pvt. Ltd.
S-9,IInd FLOOR, MLU POCKET,
MANISH ABHINAV PLAZA-II, ABOVE FEDERAL BANK,
PLOT NO-5, SECTOR -5, DWARKA, NEW DELHI, INDIA-110075,
PHONE: - + (91)-(11)-45525005