An Integrated approach of Analytical Hierarchy Process Model and Goal Model (AHP-GP Model) for Selection of Software Architecture

A Rama Mohan Reddy† **Prof. M M Naidu**†† **Prof. P.Govindarajulu**†††

SVU College of Engineering, Sri Venkateswara University, TIRUPATI, A.P, INDIA.

Abstract

Architecting the distributed software applications is a complex design activity. The selection of a best design among number of design alternatives is an important activity. To satisfy various the stakeholders' functional and non-functional requirements of a particular application, there is a need to take a number of decisions. This problem has become the multiple decision making problem. Analytical Hierarchy Process **(AHP),** integer programming, goal programming have been used in the context. In this paper we are proposing a frame work for dealing multi objective functions called an integrated approach of AHP and GOAL programming for selection of Software Architecture.

Key words:
Software Architecture, Analytical Hierarchy Process, Design Alternatives, GOAL programming

1.0 Introduction

In general the software development organizations face the problem of selecting the best design from a group of designs alternatives. Architecting the systems like distributed software is a complex design activity. It involves making decisions about a number of inter-dependent design choices that relate to a range of design concerns. Each decision requires selecting among a number of alternatives; each of which impacts differently on various quality attributes. Additionally, there are usually a number of stakeholders participating in the decision-making process with different, often conflicting, quality goals, and project constraints, such as cost and schedule. [**1**].

The basic approach of Mathematical Programming Models is to optimize the objective function while simultaneously satisfying all the constraints equations that limit the activities of the decision-maker. The current trends of research is to formulate integrated models, as the justification of problems become more complex with the identification of seemingly unconnected factors ranging from the commitment of top management and managers' perceptions towards automation to the strategic issues and production criteria such as quality, flexibility, etc. (**2**). Suresh and Kaparthi (**3**) have developed a procedure that combines a general mixed integer goal programming formulation with AHP to utilize both optimization and evaluation capabilities. A similar attempt has been made by Myint and Tabucanon **(1994)** [**4**] who effectively combined the **GP** and **AHP** methodologies for the *machine selection problem.* As a possible extension to these works on combining **AHP** and **GP** methodologies, an integrated **AHP-GP** model has been formulated for selection of software architecture design alternatives. It formally treats the priorities in the decision hierarchy of **AHP** as penalty weights of the goal constraints. This model has been applied for justifying the choice of selecting software architecture design alternatives in the case of designing the software for distributed applications. In Architecture-based and Architecture-First Software system development a few or no artifacts exist at this stage, it is hard, often impossible, to thoroughly reason about the consequences of many design decisions. Old methods evaluate and select among given coarse-grained **SAs. [5, 6, 7, 8].**

2.0 Related work

2.1 Software Architecture Evaluation Techniques

Software quality is the degree to which an application possesses the desired combination of quality attributes **[9].** Software architecture evaluation has emerged as an important software quality assurance technique. The principle objective of evaluating architecture is to assess the potential of the chosen architecture to deliver a system capable of fulfilling required quality requirements. A number of methods, such as Architecture Tradeoff Analysis Method **(ATAM) [10]** and Architecture-Level Maintainability Analysis (**ALMA) [11],** have been developed to evaluate the quality related issues at the architecture level. The architecture design evaluation methods like Quality Attribute Workshop **[12],** Cost-Benefit Analysis Method [**6**], Active Reviews for Intermediate Designs [**13**] and Attribute-Driven Design [**14**] includes a number of activities that logically belong to

Manuscript received October 5, 2007
Manuscript revised October 20, 2007

different parts of the traditional SDLC **[15].** Kazman et al**. [6]** propose the Cost Benefit Analysis Method (**CBAM**) to quantify design decision in terms of cost benefit analysis. **ATAM** is a **SA** evaluation method, which itself needs a **SA** as an input to the evaluation process. Mikael et al. **[16]** developed a quantitative approach to support the comparison of candidate architectures using Analytical Hierarchy Process (**AHP).**

2.2 The analytic hierarchy process (AHP)

The Analytic Hierarchy Process **(AHP**) is a Multi Criteria Decision Making **(MCDM**) technique that represents a complex decision problem as a hierarchy with different levels. Each level contains different elements with a relevant common characteristic. Using AHP, a cardinal measure of the importance or priority of each element in a level is obtained by pair-wise comparisons of all elements in that level. Each element in level serves as the basis for effecting pair-wise comparisons of the elements in the immediate lower level of the hierarchy. The final priorities of the elements in the lowest level (decision alternatives) are obtained using the principle of hierarchical composition. These lead to the overall ranking of design alternatives. The general methodology of **AHP** is described in detail in Saaty (**17**) and Zahedi **(18)** have made detailed reviews of its various applications. To determine the degree of quality requirements achieved in the software, the assessment of software architecture for quality is to be conducted at various phases of the software development life cycle (**SDLC**)**[19] [20**]. So Software architecture is described as various collections of architectural decisions that satisfy stake holder's choice of having multiple quality requirements **[21].** Applying AHP in a standard manner can provide overall priority weights of design alternatives. All priority weights of design alternatives can be computed using the AHP standard technique. This technique takes into consideration all quality attributes, priority weights of design alternatives for individual quality attributes and priority weights among the quality attributes themselves **[22].** In this paper, we propose Goal Programming **(GP)** techniques on results produced by the standard **AHP** for more precision in selecting the design alternative. These issues lead to an architecture better prepared for future change.

3.0 Software Architecture Evaluation

Architecture evaluation can be seen as a phase of the decision-making process. A decision-making process consists of the following activities: Problem identification; problem analysis and solution development; selection and evaluation. Though architecture evaluation focuses on selection and evaluation activities, it often covers solution development in an iterative process. Architecture evaluation allows us to forecast the optimum quality attributes by dealing with uncertainties in both subsequent implementation technology and changing requirements. Hence, we consider architecture evaluation to be a decision-making process which contains open-ended components.

Most architecture evaluation methods conduct evaluation for individual quality attributes first and consolidate the results later. Attribute-specific evaluation requires reasoning models and expertise for the quality attribute in focus. Consolidation requires a decision making process for balancing tradeoffs and selecting the best candidates when quality requirements are conflicting. To facilitate the architectural design process, Tariq Al-Naeem et al [7] proposed a quantitative quality-driven approach that attempts to find the best possible fit between conflicting stakeholders' quality goals, competing architectural concerns, and project constraints.

Architecture Tradeoff Analysis Method **(ATAM**) is a scenario-based architecture evaluation method. **ATAM** is more suitable for initially identifying trade-offs than for resolving them. If business benefits are the immediate concern of a particular architecture evaluation session and response-utility function can be solicited, **CBAM** should be used after **ATAM**. If the main concern of a particular architecture evaluation session is the overall quality (including cost if desire) of the architecture, current normal practice of **AHP** can be applied. On the other hand, it is possible to modify the current way of using **AHP** by associating weighted priority with its potential business benefits utility in the intermediate steps to enable business benefits interpretations of the final result.

3.1 AHP as a decision making tool

Analytic Hierarchy Process **(AHP),** as a critical decision making tool for several disciplines, has proved controversial **[24].** In addition, **AHP** requires users to take a holistic view of the design alternatives while comparing them without taking into account the analysis and intermediate results leading up the alternatives. This tends to neglect the Solution Development stage in a decision making process so the implications of intermediate decisions and analysis are lost. Tradeoffs with a design alternative tend to be much less explicit. This holistic view may lead to situations where the final ranking hinges on sensitive and critical decisions of which users are not aware. Several attempts **[5, 16, 7]** have been made to incorporate **AHP** into architecture evaluation.

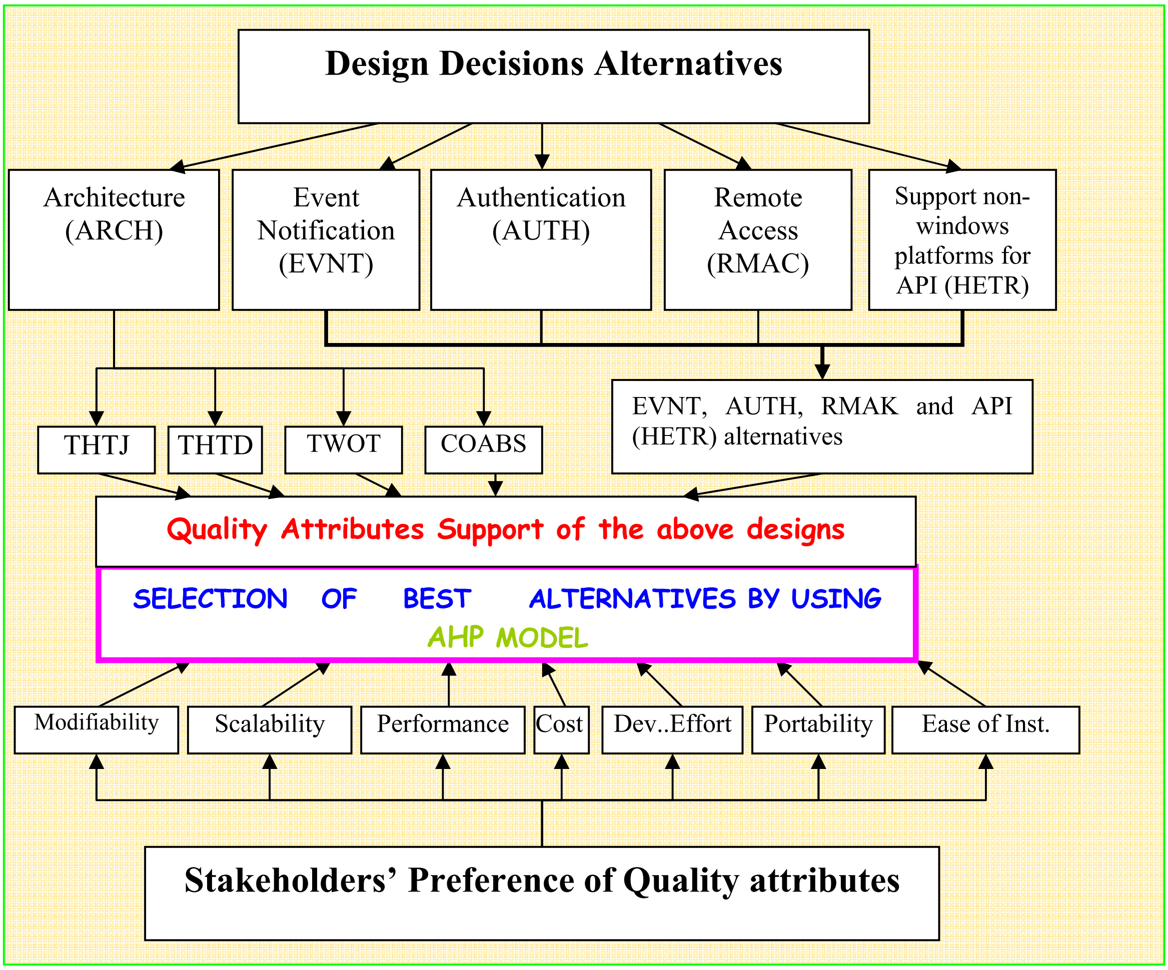

Figure 3.1 A Framework for AHP Model

Applying **AHP** in architecture evaluation is best formalized in Svahnberg et al **[16]**..

3.2 Case study: glass box project

The following is the analysis and demonstration by Al-Naeem et al **[7]** of the Glass Box Project.

The Glass Box (**GB**) project **[22**] is a part of a multi-year, research program to generate new tools and technologies for information analysts. The **GB** itself is a production software system, which is deployed in the analyst's working environment. There are approximately **15** separate research projects funded by the overall program. The research projects are required to link their software into the **GB** environment, and demonstrate their capabilities in helping analysts to solve real problems. This requires instantaneous notification of the analyst's actions, such as opening a document or performing a search. Also, in order to share knowledge generated from each research tool, there must be mechanisms for storing data generated from each tool and notifying other tools of its existence. The **figure 3.2** shows the relation ship between GB application and various stakeholders involved.

The initial **GB** version was a **2-tier client-server** system, utilizing a database, file store, and a set of tools to capture user activities when they access Web sites, document, and commenced and completed assignments. It ran standalone on each user workstations. Nightly scripts extracted the data from individual databases and emerged them into a central data store for periodic distribution to the Research Teams. Applying the principle of AHP on the **Table 2.1(Chapter 5)** the following design alternatives have been generated. The **Table 2.2** shows the final results **[7].**

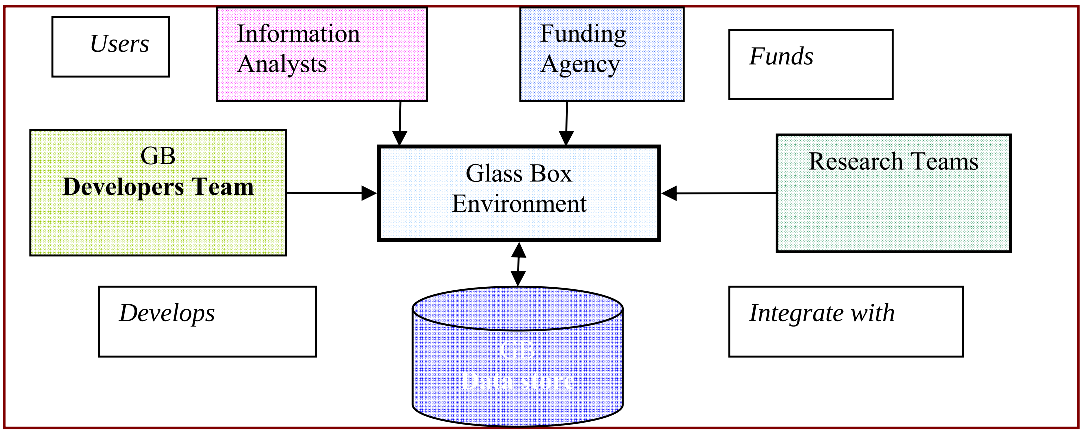

Figure 3.2 Glass Box stakeholders

3.3 The supporting quality considerations during software architecture (SA) design

Quality attributes are a central consideration during application design. Bosch **[25]** proposes a method that explicitly considers quality attributes during the design process. Hofmeister et al **[26]** describe a framework knows as global analysis to identify, accommodate, and describe architecturally significant factors including quality attributes early into the design phase. Chung et al **[27]** provides a framework that considers each design decision based on its effects on the quality attribute space.

4.0 Goal Programming

Goal Programming is concerned where a decision maker needs to consider multiple criteria in arriving at the overall best decision. Goal programming is proposed for multi-objective optimization. We consider goal programming as a function of a reference point, either a reference point with maximal objective values or an aspiration reference point.

4.1 Goal Programming Model

Goal Programming **(GP**), with many practical applications, is the most popular of all multi object decision-making techniques [**28**]. **GP** is referred to as a quantitative decision-making tool that seeks feasible solutions that achieve a certain set of desired (but adjusted) goals as closely as possibly by minimizing or penalizing deviations from the goals **(29, 30).** Another characteristic of these problems is that the objectives are apparently non-commensurable **[31].** Mathematically this problem can be represented as:

Max $[f_1(X), f_2(X) \ \ldots \ f_k(X)]$

Subject to: $g_i(X) \leq 0, (i = 1, 2\ldots m)$

Where, **X** is an n-dimensional decision variable vector. Goal Programming is one of the important methods for **MODM**. This is categorized under “Methods for a Priori Articulation of Preference Information Given”. “A Priori” means the preference information is given to the problem consists of ‘**n**’ decision variables; m is constraints and k is objectives. Any or all of the functions may be non-linear analysis before one actually solves the problem [**31**].

Charnes **&** cooper originally proposed goal Programming in **1961**. The technique has been expanded and popularized by the works of Ijiri, Lee in **1972**, [**29**] and Ignizio in **1976**. [**30**]

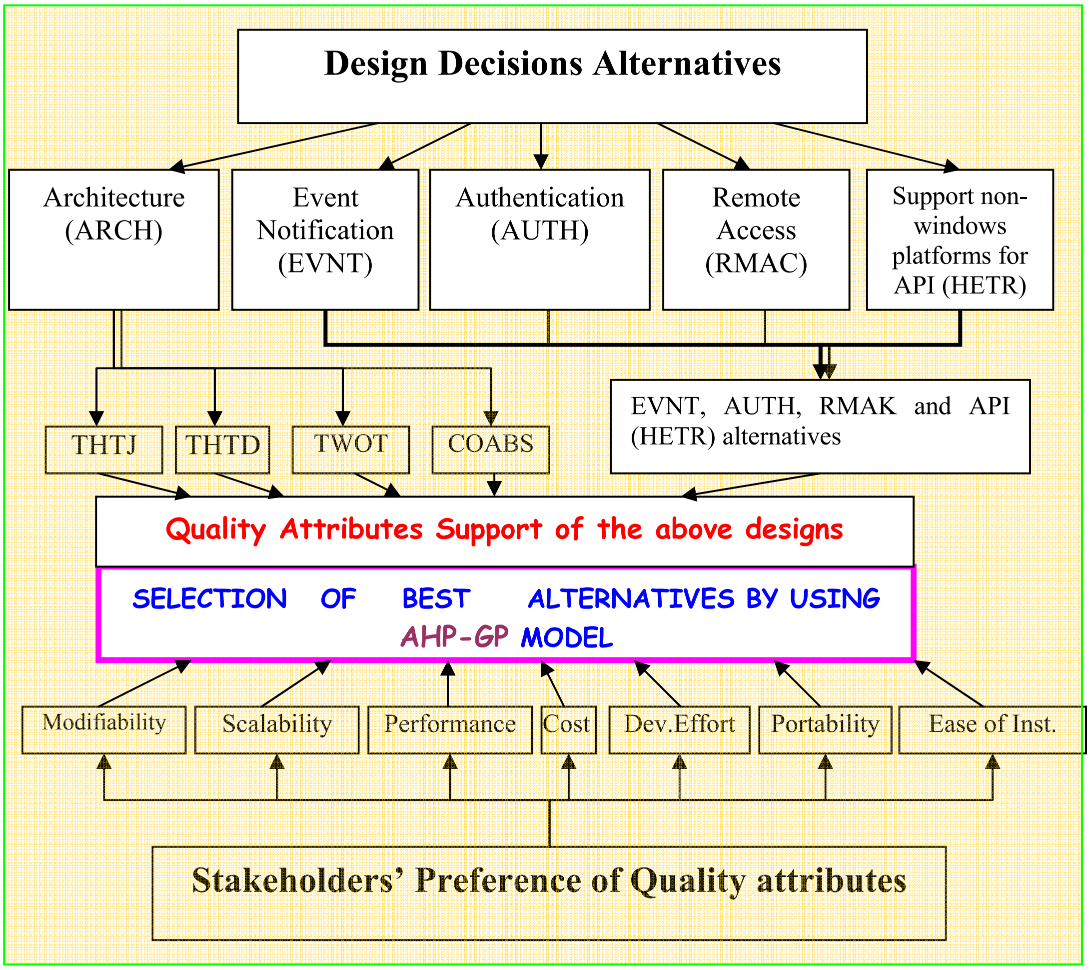

Figure 4.1 A Framework for Integrated approach to AHP-GP Model

The model is flexible enough to handle conflicting objective situations wherein only underachievement or over-achievement of goal is penalized, and conditions where the decision maker seems to come as closely as possible to a desired target. **GP** requires the assignment of ordinal priorities to the respective goals with relative weights required by any goals placed on the same priority level.

4.2 Goal Programming Formulation

The **GP** model has multi dimensional objective function that seeks to minimize certain selected absolute deviations from a stated set goals, usually within an additional set of given constraints. Each of the selected deviations in the **GP** objective carry ordinal priority weights so that goals are attained (or approached as nearly as possible) in strict order of priority. A preferred solution is then defined as the one that minimizes the deviations from the set goals. In general, the format of the GP problem can be stated as follows:

Find $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2 \ldots\ldots\ldots\ldots\ldots\ldots \ldots\ldots \mathbf{X}_n)$ so as to

Min $\mathbf{Z} = \mathbf{f}(\underline{\mathbf{d}}^{+}, \underline{\mathbf{d}}^{-}) \ldots\ldots\ldots\ldots\ldots\ldots \ldots\ldots \mathbf{(1)}$

Subject to

$\mathbf{A\ X} = \mathbf{B} + \underline{\mathbf{d}}^{+} - \underline{\mathbf{d}}^{-} \ldots\ldots\ldots\ldots\ldots\ldots \ldots.\mathbf{(2)}$

$\mathbf{C\ X} <= \mathbf{D} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ldots..\mathbf{(3)}$

$\mathbf{X}, \underline{\mathbf{d}}^{+}, \underline{\mathbf{d}}^{-} => \mathbf{0} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \mathbf{(4)}$

Where, **X** is the solution vector; Equation **(1)** is the GP objective of the problem; Equation **(2)** states the original problems objectives, converted into goals by the inclusion of intentionally permissible deviations ($\mathbf{d}_i^{+}$, $\mathbf{d}_i^{-}$) from RHS targets **($\mathbf{B}_i$);** $\mathbf{i} = \mathbf{1}, \mathbf{2} \ldots. \ldots\ldots., \mathbf{m}$ Equation (3) shows the absolute constraints on the problem; **F** $(\mathbf{d}_i^{+}, \mathbf{d}_i^{-})$ is a linear, prioritized function in of the permissible deviation variables from the associated objectives, In equation **(2)** $\mathbf{d}^{+}$ is a vector of non-negative variables that represent the permissible positive deviations from the associated objectives, in equation**(2)** **d^-** is a vector of non-negative variables, In equation**(2)**; **B** is a vector of RHS target values, of aspiration levels, associated with the objectives. In equation **(3)** **C** is a matrix of resources consumption coefficients: **A** is a matrix of activity coefficients; **D** is the vector of RHS bounds on the absolute constraints.

Quite often, the objective, Equation (**1**), takes the form:

$$\mathbf{MIN\ Z = \{\ P_1\,[\,g_1[d_1^+\ ,\ d_1^-]\,],\ P_2\,[\,g_2[d_2^+\ ,\ d_2^-]\,],\ \ldots\ldots\ldots\ldots\ldots\ldots,\ P_i\,[\,g_i[\,d_i^+\ ,\ d_i^-]\,]\ \}}$$

Where $g_i[d_i^+\ ,\ d_2^-]$, is a linear function of the deviation variables P is the ordinal priority level associated with **$g_i\ [d_i^+\ ,\ d_2^-]$, $i <= m$;** i.e., the number of ordinal priorities is equal to or less than the total number of objectives.

4.3 Deviational Variables

In GP method 'intentional' from the numerically valued goals are allowed to occur. Deviations can be either positive, negative, or zero- valued movements away from goals. All variables are non-negative in a GP model. This restriction can be circumvented by a simple transformation. For example, if **[d_i [> or = or <] 0]** is a deviation from a goal, the deviation may be replaced by

$$\mathbf{d_i = (d_i^+ - d_j^-)}$$

Where,

(a) $-\alpha < d_i < +\alpha$;

(b) $d_i^+ >= 0$;

(c) $d_i^+ >= 0$;

(d) $(d_i^+)\ (d_j^-) = 0$;

4.4 Achievement Function

The goal programming achievement function **$g_i\ [\ d_i^+,\ \ d_i^-\ ]$** is a variable that is both under the control of the decision maker and one that can have an impact on the problem solution. All decision variables are assumed non-negative.

4.5 Goal Programming For Alternative Designs

To be maximization of one of the required quality attribute, it seems logical that progress toward this global goal will be facilitated, if it is disintegrated into various sub-goals; the rational being that as the sub-goals are achieved, definite strides will be made in the direction of stakeholders' requirements maximization.

The general form of **GP** models is mathematically expressed as in **(31)**:

$$\textbf{Minimize Z} = \sum_{i=1}^{m} W_i^+ \mathbf{d_i^+} + \mathbf{Wi^- d_i^-}$$

$$\text{Subject to: } \sum_{i=1}^{m} a_{ij}\ \mathbf{x_j - d_i^+ + d_i^- = g_i \ \text{for all}\ i}$$

$$\mathbf{x_j\ ,\ d_i^+\ ,\ d_i^-\ ,\ W_i^+\ ,\ W_i^-\ >=\ 0\ \text{for all}\ i,}\ j$$

Where, W_i^+, Wj^- = pre assigned weights representing relative, pre-emptive or Combined relative-pre-emptive importance of deviations.

d_i^+ = respective positive deviations(over-achievement) from the goals: **d_i^-** = respective negative deviations (under-achievement)

from the goals:

a_{ij} = technological coefficients

x_j = decision variables,

g_i = goals

d_i^+ and d_i^- are given by the following equations;

$$\mathbf{d_i^+} = \frac{1}{2}\left[\left|\sum_{j=1}^{n} a_{ij}x_j - g_j\right| + \left(\sum_{j=1}^{n} a_{ij}x_j - g_j\right)\right]$$

$$\mathbf{d_i^-} = \frac{1}{2}\left[\left|\sum_{j=1}^{n} a_{ij}x_j - g_j\right| - \left(\sum_{j=1}^{n} a_{ij}x_j - g_j\right)\right]$$

There are at least two features of GP that need subjective inputs from decision makers: Assigning numerical weights to the objectives, and Fixing quantitative goals for the objective functions.

In addition, it is necessary to normalize the objective functions so that the deviations **$(d_i^+,\ d_i^-)$** from the goals are directly comparable. AHP has been employed by Gass **(1986) [32]** to enable decision makers to specify numerical weightages for the objectives; besides there have been other attempts to employ the Delphi technique and Conjoint analysis for this purpose. *There is a need to use **AHP** in conjunction with **GP** so as to increase the applicability of both the methodologies for problems involving syncretic (i.e., both qualitative and quantitative) criteria.* The following are some of the works that have used integrated **AHP-GP** models: (a) Ramanathan and Genesh **(1995)** [**33**] for energy resource allocation to urban households. (b) Greenberg and Nunamaker **(1994) [34]** for budgeting of public sector organizations. (c) Benjamin et al. (**1992**) [**35**] for planning facilities at the university of Missouri-Rolla, (d) Khorramshahgol et al (**1988**) [**36**] for project evaluation and selection.

5.0 AHP-GP MODEL

The integrated **AHP-GP** model has been employed to select the best architectural design alternative from a set of five alternatives for distributed applications. The Glass Box example specified the five major criteria, viz., **ARCH**, **EVNT**, **AUTHN**, **JAVA** and **HETR** as discussed in section 3.2 and [22][23][7] along with the corresponding sub-criteria.

New technology often brings operational changes, viz., flexibility, improved quality, reduced inventory, reduced work-in-progress etc., which are frequently ignored in the appraisal process. As technology justification involves active participation of different groups of specialists (stakeholders), it is absolutely necessary to have their preferences incorporated in the decision-making process. **AHP** serves as an efficient way of achieving this. The simple ranking of alternatives by using **AHP** will not adequate to completely assess the benefits of employing Architectures. A thorough analysis of the problem, by examining the levels of fulfillment of various goals (both economic and technical), is needed. GP is employed to deal with this situation. The integrated **AHP-GP** model provides an excellent means to combine design decisions with the choice of technological alternatives available. Data have been collect pertaining to the criteria identified in the Glass-Box project [7][37]. AHP has been used to set priorities among the nineteen(19) sub-criteria (belonging to the five major criteria) identified earlier. The stakeholders gave subjective value judgments[7], which were used, in the pair wise comparison matrices. The computational details of AHP have been provided for the levels **1** and **2** in **Tables 5.1** and **5. 2**.

The architecture (**ARCH**) criteria received the maximum priority followed by Event Notification (**EVNT**), Security (**SECU**) Authentication (**AUTH**), and **HETR**. The priority weights of the individual sub-criteria have also been indicated in **Figure 5.2, 5.3, 5.4, 5.5, 5.6 and 5.7**. These priority weights are then used as penalty weights while formulating the **GP** model. They have been converted into percentages before being employed as penalty weights.

The complete **GP** formulation has been presented subsequently. The utility and effectiveness of the integrated **AHP-GP** model for justifying the choice of Architectural design alternatives have been given in this case study. Integration of **ARCH, SECUR, EVNT, AUTH**, and **HETR** criteria which involve both quantitative and qualitative sub-criteria. A comprehensive analysis of the problem, by taking into account the ratings and opinions of different stakeholders involved in the project. Incorporation of multiple conflicting objects that do not necessarily have to be commensurable.

5. 10 PREFERENCES OF STAKEHOLDERS ON ALTERNATIVE DECISIONS GLASS – BOX EXAMPLE

Table 5.1 **Weights of the different design decisions across a 10-point weighting scale.**

Sl.No.	Design Decision	Preferences of Lead Architect
1.	Architecture (ARCH)	**10**
2.	Event Notification (EVNT)	**8**
3.	Authentication (AUTH)	**1**
4.	Remote Access (RMAC)	**4**
5.	Supporting Non-Windows Platforms API(HETR)	**5**

Table 5.2 AHP WEIGHTS for Design Decision Alternatives

Design Decision Alternatives	Architecture (ARCH	Event notification (EVNT)	Authentica-tion (AUTH)	Remote Access (RMAC)	API (HETR)	AHP WEIG HTS
Architecture (ARCH)	1 1.0000	2 2.0000	9 9.0000	6 6.0000	5 5.0000	0.482
Event notification (EVNT)	½ 0.5000	1 1.0000	7 7.0000	4 4.0000	3 3.0000	0.287
Authentication (AUTH)	1/9 0.1111	1/7 0.1428	1 1.0000	1/3 0.3333	¼ 0.25	0.037
Remote Access (RMAC)	1/6 0.1666	¼ 0.25	3 3.0000	1 1.0000	1 1.0000	0.090
API (HETR)	1/5 0.2000	1/3 0.3333	4 4.0000	1 1.0000	1 1.0000	0.105

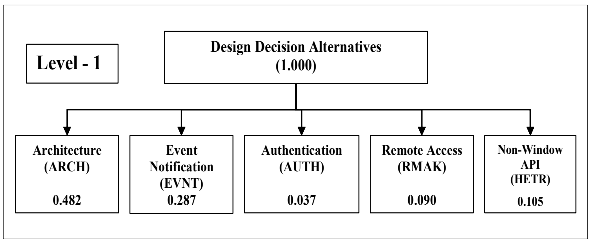

Figure 5.2 **AHP weights for** Design Decision Alternatives

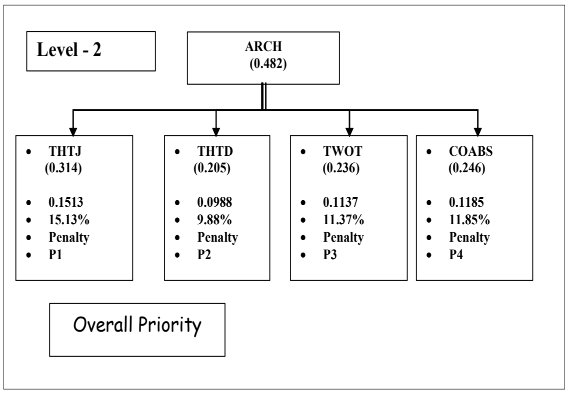

Figure 5. 3 AHP weights for Architecture Decisions

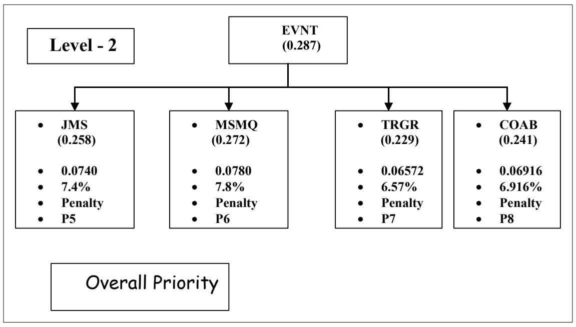

Figure 5.4 Event Generation alternatives

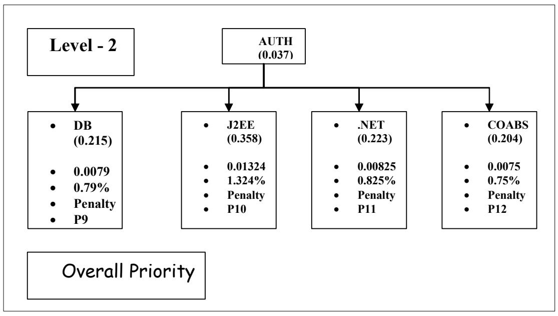

Figure 5.5 AHP weights for Authentication Alternatives

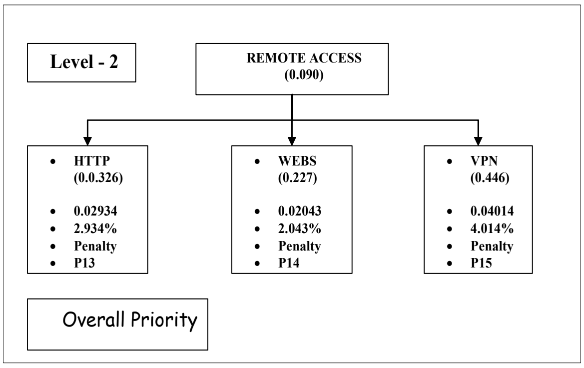

Figure 5.6. Weights of REMOTE CRITERION decision alternatives

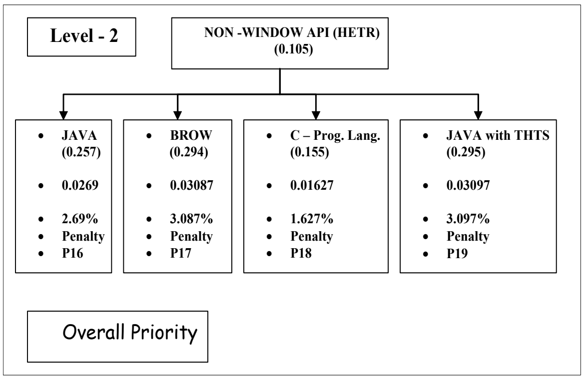

Figure 5.7 AHP weights for Non-Window API (HETR)

5.1 GOAL PROGRAMMING: Achievement Function

ACHIEVEMENT FUNCTION:

Maximize

$Z = P1\,(d_1^{+} + d_2^{-}) + P2\,(d_3^{+} + d_4^{-}) + P3\,(d_5^{+} + d_6^{-})$
$+ P4\,(d_7^{+} + d_8^{-}) + P5\,(d_9^{+} + d_{10}^{-}) + P6\,(d_{11}^{+} + d_{12}^{-})$
$+ P7\,(d_{13}^{+} + d_{14}^{-}) + P8\,(d_{15}^{+} + d_{16}^{-}) + P9\,(d_{17}^{+} + d_{18}^{-})$
$+ P10\,(d_{19}^{+} + d_{20}^{-}) + P11\,(d_{21}^{+} + d_{22}^{-}) + P12\,(d_{23}^{+} + d_{24}^{-})$
$+ P13\,(d_{25}^{+} + d_{26}^{-}) + P14\,(d_{27}^{+} + d_{28}^{-}) + P15\,(d_{29}^{+} + d_{30}^{-})$
$+ P16\,(d_{31}^{+} + d_{32}^{-}) + P17\,(d_{33}^{+} + d_{34}^{-}) + P18\,(d_{35}^{+} + d_{36}^{-})$
$+ P19\,(d_{37}^{+} + d_{38}^{-})$

OBJECTIVES:

(a) ARCH – THTJ design alternative (P1):
$\sum j_i x_i - d_k^{+} + d_k^{-} = J$
Where,
ji = THTJ design alternative,
xi = decision variable,
dk = deviation variable
J = ARCH – THTJ design alternative goal to be achieved.

(b) ARCH – THRD design alternative (P2):
$\sum d_i x_i - d^{+}_k + d_k^{-} = D$
di = THTD design alternative,
D = ARCH – THTD design alternative goal to be achieved

(c) ARCH – TWOT design alternative (P3):
$\sum t_i x_i - d_k^{+} + d_k^{-} = T$
t_i = THTD design alternative
T = ARCH-TWOT design alternative goal to be achieved.

(d) ARCH – COABS design alternative (P4):
$\sum c_i x_i - d_k^{+} + d_k^{-} = C$
c_i = COABS design alternative
C = ARCH-COABS design alternative goal to be achieved.

(e) EVNT – JMS design alternative (P5):
$\sum m_i xi - d_k^{+} + d_k^{-} = M$
m_i = JMS design alternative
M = EVNT-JMS design alternative goal to be achieved

(f) EVNT- MSMQ design alternative (P6):
$\sum q_i xi - d_k^{+} + d_k^{-} = Q$
Qi = MSMQ design alternative
Q = EVNT-MSMQdesign alternative goal to be achieved

(g) EVNT – TRGR design alternative (P7):
$\sum g_i x_i - d_k^{+} + d_k^{-} = G$
g_i = TRGR design alternative
T= EVNT–TRGR design Alternative goal to be achieved

(h) EVNT – COABS design alternative (P8):
$\sum b_i x_i + - d_k^{+} + d_k^{-} = B$

b_i = COABS design alternative
B = EVNT – COABS design alternative goal to be achieved

(i) AUTH-DB design alternative (P9):
$\sum a_i x_i - d_k^+ + d_k^- = A$
a_i = DB design alternative
A = AUTH-DB design alternative goal to be achieved

(j) AUTH –J2EE design alternative (P10):
$\sum e_i xi - d_k^+ + d_k^- = E$
e_i = J2EE design alternative
E = AUTH –J2EE design alternative goal to be achieved

(k) AUTH- .NET design alternative (P11):
$\sum n_i x_i - d_k^+ + d_k^- = N$
n_i = THTD design alternative
N = AUTH- .NET design alternative goal to be achieved

(l) AUTH – COABS design alternative (P12):
$\sum o_i x_i - d_k^+ + d_k^- = O$
o_i = COABS design alternative
T = AUTH – COABS design alternative goal to be achieved

(m) REMOTE ACCESS- HTTP design alternative (P13):
$\sum h_i x_i - d_k^+ + d_k^- = H$
h_i = HTTP design alternative
H = REMOTE ACCESS- HTTP design alternative goal to be achieved

(n) REMOTE ACCESS- WEBS design alternative (P14):
$\sum w_i x_i - d_k^+ + d_k^- = W$
w_i = WEBS design alternative
W = REMOTE ACCESS- WEBS design alternative goal to be achieved

(o) REMOTE ACCESS- VPN design alternative (P15):
$\sum v_i x_i - d_k^+ + d_k^- = V$
v_i = VPN design alternative
V = REMOTE ACCESS- VPN design alternative goal to be achieved

(p) API(HETR)-JAVA design alternative (P16):
$\sum r_i x_i - d_k^+ + d_k^- = R$
r_i = THTD design alternative
R = API(HETR)-JAVA design alternative goal to be achieved

(q) API (HETR)- BROW design alternative (P17):
$\sum u_i x_i - d_k^+ + d_k^- = U$
u_i = THTD design alternative
U = API (HETR)- BROW design alternative goal to be achieved

(r) API (HETR) - C-Prog. Lang. Design alternative (P18):
$\sum l_i x_i - d_k^+ + d_k^- = L$
l_i = C-Prog. Lang design alternative
L = API (HETR) - C-Prog. Lang. design alternative goal to be achieved

(s) API (HETR) - JAVA with THTS design alternative (P19):
$\sum z_i x_i - d_k^+ + d_k^- = Z$
z_i = JAVA with THTS design alternative
Z = API (HETR) - JAVA with THTS design alternative goal to be achieved.

6.0 Conclusion

As *technology justification involves active participation of different groups of specialists (stakeholders), it is absolutely necessary to have their preferences incorporated in the decision-making process*. The integrated **AHP-GP** model provides an excellent means to ***combine design decisions with the choice of technological alternatives available.*** Data have been collect pertaining to the criteria identified in the Glass-Box project. The then stakeholders subjective value judgments, which were used, in the pair wise comparison matrices. The computational details of AHP have been provided for the levels **1** and **2** in **Table 5. 1** and **5.2**. *AHP-GP model formally treats the priorities in the decision hierarchy of AHP as penalty weights of the goal constraints.* This model has been applied for justifying the choice of selecting software architecture design alternatives in the case of designing the software for distributed applications. These issues lead to an architecture better prepared for future change.

7.0 Reference

[1] Chung L et al, “Non-Functional Requirements in Software Engineering”: Kluwer Academic Publishers, Boston, MA. 1999].

[2] Chen, Frank F. and Everett E. Adam Jr., 1991. “The impact of flexible manufacturing systems on productivity and quality”. IEEE Transactions on Engineering Management, Vol.38, No.1, pp.33-45.

[3] Suresh and Nallan C, and Shashidhar Kaparthi, 1992. “Flexible automation investments: A synthesis of two multi-objective modeling approaches”. Computer and Industrial Engineering, Vol.22, No.3, pp. 257-272.

[4] Myint, S and M.T. Tabucanon, 1994. “A multi-criteria approach to machine selection for flexile manufacturing systems”. International journal of Production Economics, Vol.33, Nos.1-3, pp.121-131.

[5] Svahnberg, M., Wohlin, C., Lundberg, L., and Mattsson, M, 2002. “A Method for understanding quality attributes in software architecture structures”, In Proceedings of the 14th international conference on Software Engineering and Knowledge engineering (SEKE), pp. 819-826.

[6] Kazman, Rick; Asundi, Jai; & Klein, Mark. “Quantifying the Costs and Benefits of Architectural Decisions,” 297-306. Proceedings of the 23rd International Conference on Software Engineering (ICSE 2001). Toronto, Ontario, Canada, May 12 - 19, 2001. Los Alamitos, CA: IEEE Computer Society, 2001.

[7] Al-Naeem T et al., “A Quality-Driven Systematic Approach for Architecting Distributed Software Applications”. UNSW-CSE-TR-0433. 2004.

[8] Roy T. Fielding. “Software Architectural Styles for Network-Based Applications”. University of California, Irvine. Phase II Survey, 1999.

[9] Parnas D. L. “On the criteria to be used in decomposing systems into modules”. Communications of the ACM, 15(12), Dec, 1972, pp. 339-344.

[10] Kazman, R., Barbacci, M., Klein, M., and Carriere, J 1999. “Experience with performing architecture tradeoff analysis”. In Proceedings of the 21st International Conferences on software Engineering (ICSE’99), pp.54-63.

[11] Bengtsson, P., Lassing. N., Bosch, J., and Vliet, H.v. 2004. “Architecture-level modifiability analysis (ALMA)”. Journal of Systems and Software 69(1/2): 129-147.

[12] Barbacci, Mario R.; Ellison, Robert; Lattanze, Anthony J.; Stafford, Judith A.; Weinstock, Charles B.; & Wood, William G. “Quality Attribute Workshops”, Third Edition (CMU/SEI-2003-TR- 016). Pittsburgh, PA.

[13] Clements, Paul C. “Active Reviews for Intermediate Designs” (CMU/SEI-2000-TN-009, ADA383775). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.

[14] Bass, Len; Clements, Paul; & Kazman, Rick. “Software Architecture in Practice”, Second Edition. Boston, MA: Addison-Wesley, 2003.

[15] Kazman Rick “A Life-Cycle View of Architecture Analysis and Design Methods”, September 2003, TECHNICAL NOTE, CMU/SEI-2003-TN-026

[16] Svahnberg M, C. Wholin, and L. Lundberg, “A Quality-Driven Decision-Support Method for Identifying Software Architecture Candidates”. Int. Journal of Software Engineering and Knowledge Engineering, 2003, 13(5):p. 547-573

[17] Saaty, Thomas L., 1980. “The Analytic Hierarchy Process”. McGraw-Hill, New York

[18] Zahedi, Fatemah, 1986, “The Analytic Hierarchy Process: A Survey of the method and its applications”, Interfaces, Vol. 16, pp.96-108.

[19] Perry D. E and A. L Wolf. “Foundations for the study of software architecture”. ACM SIGSOFT Software Engineering Notes, 17(4), Oct. 1992, pp.40-52

[20] Bosch, J. 2004., “Software Architecture: The next step”, In Proceedings of the First European Workshop on Software architecture (EWSA), pp. 194-199.

[21] ZHU et al. “Tradeoff and Sensitivity Analysis in Software Architecture Evaluation Using Analytic Hierarchy Process”, 2005.

[22] Greenberg, Robert R, and Thomas R Nunamaker, 1994. “Integrating Analytic Hierarchy Process (AHP) into the multi objective budgeting models of public sector organizations”. Socio-Economic Planning Sciences, Vol.28 No.3, pp.197-206.

[23] Triantaphyllou, E. and Mann, S.H. 1994. “Some critical issues in making decisions with pair-wise comparisons”. In Proceedings of the Third International Symposium on the AHP, pp.225-236.

[24] Bosch J, “Design & Use of Software Architectures: Adopting and evolving a product line approach”: Addison-Wesley, 2000.

[25] Hofmeister C, R. L Nord, and D. Soni, “Applied Software Architecture”. Teading, MA: Addison-Wesley Longman, 2000.

[26]

[27] White D. J., 1990. “A bibliography on the application of mathematical programming multiple-objective methods”, Journal of Operational Research society, Vol.41, No. 8, pp.669-691.

[28] Lee, Sang M., 1972. “Goal Programming for Decision Analysis”, Auerbach Publishers, Philadelphia.

[29] Ignizio, James P. “Goal Programming and Extensions”. Lexington Books, D.C.Heath and Company, Massachusetts, 1976.

[30] Hwang Ching –Lai and Abu Syed Md. Masul, Hutti, “Objective decision Making Methods and Applications”, New York; Springer - Verlag, 1997.

[31] Min, Hokey and James Storbeck, 1991. “On the origin and persistence of misconceptions in goal programming”. Journal of Operational Research Society, Vol. 42, No.4, pp.301-312

[32] Gass, Saul I., 1986. “A Process for determining priorities and weights for large scale linear goal programmes”. Journal of Development Studies, Vol.21, No.4, pp.572-652.

[33] Ramanathan R and L. S Genesh, 1995. “Energy resource allocation incorporating qualitative and quantitative criteria: An Integrated model using goal programming AHP”. Socio-Economic Planning Sciences, Vol.29, No.3, pp.197-218.

[34] Benjamin, Colin O., Ike C. Ehie, and Yildirim Omurtag, 1992. “Planning facilities” at the University of Missouri-Rolla. Interfaces, Vol.22, No. 4, pp.95-105.

[35] Khorramshahgol, Reza, Hussein Azani, and Yvan Gousty, 1988. “An integrated approach to project evaluation and selection”. IEEE Transactions on Engineering Management, Vol.35, No.4, pp.265-270.

[36] Gorton I, and J. Haack. “Architecting in the Face of Uncertainty”: An Experience Report. Proc. International Conference on Software Engineering. 2004. Edinburgh, Scotland.

[37] Francisca Losacio and ledis chirinos Nicole Levy and Amar Ramdane-cherif,” Quality Characteristics for software Architecture”, Journal of Object Technology, vol.2, No.2, March-April 2003.

A Rama Mohan Reddy received the B.Tech. from JNT University, Hyderabad, India, in 1986 and M.Tech degrees in Computer Science from National Institute of Technology in 2000 Warangal, India. He worked As Assistant Professor, and Associate Professor of Computer Science and Engineering, Sri Venkateswara University College of Engineering during the period 1992 and 2005. Presently working as Professor and Head of Department of Computer Science and Engineering, Sree Vidyanikethan Engineering College.