An Analytical Approach for Reliability Analysis of Pipeline Software Architecture

Swapna S. Gokhale ¹ Sherif Yacoub ²

¹ Dept. of Computer Science and Engineering
Univ. of Connecticut
Storrs, CT 06269, USA
Email: ssg@engr.uconn.edu

² Hewlett-Packard Labs, Barcelona Research Office
Barcelona, Spain
Email: sheriff.yacoub@hp.com

Abstract: Architecture styles represent commonly occurring patterns of the structural organization of components and connectors of an application. A number of such styles have been identified and analyzed extensively for different non functional attributes including performance, maintainability, flexibility, and modifiability. The issue of reliability analysis of architecture styles, however, has been relatively less investigated. This paper presents a reliability analysis methodology for one such architecture style, namely, the pipe and filter style. Two variants of the topological organization of the pipes and filters, namely, linear topology without feedback and with feedback where the feedback loop is incorporated either to improve quality or to improve reliability are considered. The methodology derives analytical expressions for application reliability which incorporate the impact of (i) error propagation and downstream error correction, and (ii) deterministic number of iterations through the feedback loop, with filter reliabilities a function of the number of iterations. The potential of the methodology to obtain a reliability estimate and to facilitate sensitivity analysis is illustrated using an industrial case study of a Document Understanding and Analysis Application.

Keywords: Software architecture, Software reliability, Pipeline style, Feedback loop.

1. Introduction

Software architecture is receiving increasing appreciation as a critical design level for software systems, as they continue to grow in size and complexity. Software architecture is concerned with many aspects such as high-level component organization, protocols for communication, synchronization and data access, global control structures, performance, assignment of functionality to design elements, and the selection amongst design alternatives [12]. Software architecture represents the design decisions that are made in the early phases of a system and these decisions are usually difficult to change or reverse. These architectural choices have a profound influence on the non functional attributes that can be supported by a system. Software architecture analysis can be used to assess the degree to which a given architecture supports important quality attributes such as maintainability, reliability, reusability, and performance. Many different techniques have appeared for the analysis of software architectures for their non functional attributes such as performance [16] [29], maintainability [8], evolution and reusability [22], flexibility [20], modifiability [9], and reliability [14] [15] [17] [32].

Over the years, some commonly occurring patterns of the structural organization of components and connectors have been identified, and these patterns are referred to as architecture styles. Examples of architecture styles include: pipe and filter, event-based, implicit invocation, etc. [12]. These styles differ from one another in the way control and data are handled. They have been analyzed for different attributes including performance, maintainability, and modifiability. Little attention has been provided, however, towards the reliability analysis of architecture styles [3] [28].

This paper presents a reliability analysis methodology for one such architecture style, namely, the pipe and filter style. Two variants of the topological organization of the pipes and filters, namely, linear topology without feedback and linear topology with feedback where the feedback loop is incorporated either to improve quality or to improve reliability are considered. The methodology incorporates the impact of (i) error propagation and downstream error correction, and (ii) deterministic number of iterations through the feedback loop, with filter reliabilities a function of the number of iterations. Closed form, analytical expressions which relate application reliability to the reliabilities of the pipes and filters and the parameters of the feedback loop are derived. The potential of the methodology to obtain an estimate of the application reliability and to facilitate sensitivity analysis is illustrated using an industrial case study of a Document Analysis and Understanding Application.

The layout of the paper is as follows: Section 2 provides an overview of the pipe and filter style. Section 3 describes the reliability analysis methodology. Section 4 illustrates the methodology with an industrial case study. Section 5 summarizes related research. Section 6 offers concluding remarks and directions for future research.

2. Overview of pipe and filter style

In the pipe and filter style, each component reads a stream of data on its input and produces a stream of data on its output. Input is transformed both locally and incrementally so that output begins before the entire input stream is consumed.
Components are termed filters; connectors serve as conduits for the information streams and are termed pipes. The pipes and filters may be connected to form a generic topology as shown in Fig. 1. The main characteristics of this style include the condition that filters must be independent entities, and they need not know the identities of the upstream and the downstream filters. They may specify the input format and guarantee what appears on the output, but they may not know which components appear at the end of those pipes.

Though the pipe and filter architecture style appears structurally simple, it is used as the basic underlying style of many applications because it possesses many advantages [12]. Filters are usually stand alone and can be treated as black boxes that encapsulate specific functionality. This encapsulation helps ensure quality attributes such as information hiding, low coupling, high cohesion, and reuse. The pipe and filter style is used in many critical, practical applications, e.g. operating systems (Unix pipes and filters) and compilers (laxer, parser, semantic analyzer, code generation). It is also used in applications that are heavily based on data flow between concurrent, mostly independent processes. More recently, the pipe and filter style is also used in applications which provide messaging services for communication among different entities in service-oriented architectures [11]. For all such applications, reliability is of paramount importance.

3. Reliability analysis methodology

The reliability analysis methodology is presented in this section. First, details of the topological organization of the pipes and filters, followed by the purpose and the operational logistics of the feedback loop are presented. The processing policies and input/output relationships for the filters are then discussed. Finally, derivations of the analytical reliability expressions are presented.

3.1 Topological organization

A linear topology of $n$ filters, labeled sequentially from $1$ through $n$ from the left is considered. The filters are connected by pipes, each pipe is labeled $(a, a+1)$, where $a$ ranges from $1$ through $n-1$. Two cases of the linear topology are chosen. In the first case, termed as “linear topology without feedback”, each data element is processed sequentially from the first filter through the $n^{th}$ filter. Fig. 2 shows the pipes and filters organized into a linear topology without feedback. The second case includes a feedback loop in the topology and is termed as “linear topology with feedback”. The feedback loop covers filters $n_2$ through $n_2$ as in Fig. 3. Thus, each element of input data is processed by filters $1$ through $n_2-1$ and $n_2+1$ through $n$ just once, and may be processed multiple times by filters $n_1$ through $n_2$.

Fig. 1: Generic topology of pipes and filters.

Fig. 2: Linear topology of pipes and filters without feedback.

Fig. 3: Linear topology of pipes and filters with feedback.

3.2 Feedback loop

A feedback loop may be incorporated into the topology for one of two purposes. The first purpose is to improve the quality of the output and the second purpose is to improve the reliability of the processing. In both these cases, additional checks are employed to assess the intermediate results obtained after the processing by filter $n_2$ is completed. This check first determines whether a failure has occurred after processing is completed by filter $n_2$. If no failure has occurred and if the feedback loop is incorporated to improve quality then a further analysis is used to measure and to determine if the quality of the output is acceptable according to a pre-specified criterion. The criterion to determine whether the quality is acceptable or not is specific to the application. For example, in the case of a document transformation system (see Section 4), the final output can be checked for “look and feel” and to ensure that no part of a page has been cut off by the processing algorithms. In the case of an image processing system, the output can be checked for acceptable resolution of the image. If the output does not satisfy the desired quality concerns, it is discarded and the data element is subject to another round of processing by filters $n_1$ through $n_2$. This process is repeated until either the quality of the output obtained from filter $n_2$ is satisfactory or a certain maximum number of iterations, denoted $m$ are completed. In each round of processing, a modified or other feed forward algorithm is applied to improve the quality. In addition, the components used may be reconfigured, tuned, or replaced to increase the possibility of obtaining an acceptable result. The maximum number of iterations $m$ is determined by the number of possible configurations of the filters that can be explored to improve quality. The output from each feedback iteration is retained until a decision to break from the loop is reached, after which the best output from the history of iterations is selected. Let $q_r$ denote the probability that the output obtained from filter $n_2$ in the $r^{th}$ iteration satisfies the quality criteria. In this case, each time the data element is processed by filters $n_1$ through $n_2$ to improve quality, there is a risk that the filters may fail. However, since the element was processed correctly in the previous iterations, the failure probability decreases for each subsequent iteration. Thus, the probability of processing an element correctly for filters $n_1$ through $n_2$ is an increasing function of the number of iteration.

If the feedback loop has been included to improve reliability, and if no failure has occurred after the processing of filter $n_2$, then the output of filter $n_2$ is fed into the downstream filter $n_2+1$. However, if a failure has occurred upon the processing of filter $n_2$, then the data is subjected to another round of processing by filters $n_1$ through $n_2$. This process is repeated until either the output of filter $n_2$ is correct.
or a certain maximum number of processing iterations denoted $m$ is reached. A different feed forward algorithm or a configuration option is tried in each iteration to improve the possibility that the data element is processed without failure. Unlike the case where the feedback loop is incorporated to improve quality, the probability of failure for filters $n_1$ through $n_2$ is the same in each iteration.

### 3.3 Processing policies

Each filter receives one element of data at a time and it processes that element. The input to the first filter is provided by an external source, while the inputs to filters 2 through $n$ are provided by the outputs of the upstream filters 1 through $n-1$ respectively. These input/output relationships hold even for the topology with a feedback loop. In this case, if additional rounds of processing by filters $n_1$, through $n_2$ are necessary, then the input to filter $n_j$ in these additional rounds is received from filter $n_j-1$ and not from filter $n_2$. Thus, the feedback loop is a control loop and not a data loop.

The processing of each element of input data by a filter yields three possibilities: it produces a correct result, it produces an incorrect result, or it does not produce any output. Which one of these three possibilities is considered as a failure of the filter depends on the policy that is employed. Two policies, namely, conservative and opportunistic are considered. In the conservative policy, when a filter produces an incorrect result or no result, it is considered to have failed. In this case, the incorrect result is not transmitted via the pipe to the downstream filter. On the other hand, in the opportunistic policy, a filter is considered to have failed only when it produces no output. If a filter produces an incorrect output, it is passed on to the subsequent downstream filter for further processing. Thus, in this policy the downstream filters have the opportunity to correct an error committed by an upstream filter and to provide correct output despite the fact that one or more of the filters may have produced an incorrect result. Due to this additional opportunity to remedy the failure(s), this policy is termed opportunistic. Assuming that the external input is always correct, in the conservative policy, the inputs received by each filter are always correct. On the other hand, in the opportunistic policy, the inputs received by each filter except for the first one, can be either correct or incorrect.

A total of six possible combinations arise based on the different types of inputs received and outputs produced by each filter. For filter $j$, the notation for conditional probability for each of these combinations is provided in Table 1. Further, the unconditional or the absolute probabilities for filter $j$ receiving different types of inputs and producing different types of outputs are given in Table 2.

### Table 1: Conditional probabilities of filter $j$.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Conditional Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_j(C)$</td>
<td>Correct output, given correct input.</td>
</tr>
<tr>
<td>$p_j(I)$</td>
<td>Correct output, given incorrect input.</td>
</tr>
<tr>
<td>$p_j(N)$</td>
<td>Incorrect output, given correct input.</td>
</tr>
<tr>
<td>$p_j(I')$</td>
<td>Incorrect output, given incorrect input.</td>
</tr>
<tr>
<td>$p_j(N')$</td>
<td>No output, given correct input.</td>
</tr>
<tr>
<td>$p_j(N'')$</td>
<td>No output, given incorrect input.</td>
</tr>
</tbody>
</table>

### Table 2: Absolute/probabilities of filter $j$.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Absolute Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_j(C)$</td>
<td>Correct output.</td>
</tr>
<tr>
<td>$p_j(I)$</td>
<td>Incorrect output.</td>
</tr>
<tr>
<td>$p_j(N)$</td>
<td>No output.</td>
</tr>
<tr>
<td>$I_j(I)$</td>
<td>Incorrect input.</td>
</tr>
<tr>
<td>$I_j(C)$</td>
<td>Correct input.</td>
</tr>
</tbody>
</table>

Using the theorem of total probability [27], for filter $j$, the unconditional or absolute probabilities can be obtained from the conditional probabilities. The absolute probability of producing correct output $p_j(C)$ is given by:

$$p_j(C) = I_j(C)p_j(C \mid C) + I_j(I)p_j(I \mid I)$$  \hspace{1cm} (1)

Similarly, the absolute or total probabilities of producing incorrect and no output are given by:

$$p_j(I) = I_j(C)p_j(I \mid C) + I_j(I)p_j(I \mid I)$$  \hspace{1cm} (2)

$$p_j(N) = I_j(C)p_j(N \mid C) + I_j(I)p_j(N \mid I)$$  \hspace{1cm} (3)

The methodology considers the following four variants of the pipeline architecture based on the combination of the topological organization and processing policy: (i) topology without feedback and conservative policy, (ii) topology without feedback and opportunistic policy, (iii) topology with feedback for quality improvement and conservative policy, and (iv) topology with feedback for reliability improvement and conservative policy.

### 3.4 Filter and pipe reliabilities

The reliability of a filter is the probability that it processes an element of data correctly. In the conservative policy, since only the case of correct output is regarded as success, filter reliability is given by $p_j(C)$. Further, since the input received by each filter is always correct, $p_j(C) = p(C)$ for all the filters. In the opportunistic policy, since the cases of correct and incorrect output are regarded as success, filter reliability is given by $p_j(C) + p_j(C'I)$. In the linear topology without feedback, the reliability of each filter is the same each time it processes an element of data. This also holds for linear topology with feedback, when the feedback is used to improve reliability, since the reliabilities of filters $n_1$, through $n_2$ in the feedback loop do not change with the number of processing iteration. On the other hand, however, when the feedback loop is used to improve quality, the reliabilities of filters $n_1$, through $n_2$ are a function of the number of processing iteration as discussed in Section 3.2. Since only conservative policy is employed in conjunction with the feedback loop, filter reliabilities are given by $p_j(C)$. Let $p_j(V,C)$ denote the iteration-dependent probability of producing a correct output, where $j$ ranges from $n_1$ through $n_2$ for filters involved in the feedback loop and $V$ is the number of iteration. The considerations in determining the form of $p_j(V,C)$ as a function of the number of processing iteration are as follows. In each iteration, the probability of producing correct output should increase compared to the previous one, by the virtue of the fact that the data element was processed correctly in all the prior iterations. This probability, however, should never exceed
unity. Secondly, the rate at which this probability increases should decrease as the number of iterations increase. In other words, the incremental reliability improvement in going from the first iteration to the second one should be less than the incremental improvement in going from the second iteration to the third one and so on. The following functional form of \( p_j(V,C) \) satisfies the above considerations:

\[
p_j(V,C) = p_j(C)^{1/d}
\]  

(4)

In Equation (4) the parameter \( d \) determines the rate at which \( p_j(V,C) \) increases and approaches unity. The parameter \( d \) should be adjusted so that \( p_j(V,C) \) approaches unity within the desired number of iterations to indicate that beyond this number of processing iterations the probability of failure is negligible. \( p_j(C) \) is the probability of producing correct output in the first iteration. Figs. 4 through 7 show the variation of \( p_j(V,C) \) as a function of \( p_j(C) \) and \( d \). Four values of \( p_j(C) \), namely, 0.6, 0.7, 0.8 and 0.9 and three values of \( d \), namely, 1, 2 and 3 were considered. The figures indicate that the higher the values of \( p_j(C) \) and \( d \), the lower is the number of iterations at which the failure probability becomes negligible.

A pipe does not process and modify data; it only serves as a vehicle to transport data. Thus, the failure of the pipe implies unavailability of the channel used for the transport of data. Let \( p_{a,b} \) denote the probability that the pipe \((a,b)\) is available to transport a single element of data.

3.5 Reliability expressions

The analytical reliability expressions for the four variants of the pipe and filter style identified in Section 3.3 are derived in this section. The application reliability is defined as the probability of processing an element of data correctly.

Case 1: Topology without feedback, conservative policy

In this case, since the incorrect output of each filter is regarded as a failure and is not forwarded to the downstream filter for further processing, for each filter including the first one \( I_j(I) = 0.0, I_j(C) = 1.0 \). Thus, from Eq.(1), Eq.(2) and Eq.(3) respectively, \( p_j(C) = p_j(C|C) \), \( p_j(I) = p_j(I|C) \), and \( p_j(N) = p_j(N|C) \). The application reliability is given by:

\[
R = \left( \prod_{i=1}^{n} p_i(C) \right) \left( \prod_{i=1}^{n-1} p_{i+1}(I) \right)
\]

(5)

Case 2: Topology without feedback, opportunistic policy

In this case, filter \( j (j > 1) \) receives both incorrect and correct inputs, when the outputs produced by its upstream filter \( j-1 \) are incorrect and correct respectively. Thus, based on the probabilities of correct and incorrect outputs of filter \( j-1 \), namely, \( p_{j-1}(C) \) and \( p_{j-1}(I) \), \( I_j(C) \) and \( I_j(I) \) can be obtained from the following expressions:

\[
I_j(C) = \frac{p_{j-1}(C)}{p_{j-1}(C) + p_{j-1}(I)}
\]

(6)

\[
I_j(I) = \frac{p_{j-1}(I)}{p_{j-1}(C) + p_{j-1}(I)}
\]

(7)
The normalization in Eq. (6) and Eq. (7) is necessary since one of the three types of output produced by filter j-1, only two are fed into filter j for subsequent processing. When filter j-1 produces no output, no input is fed into the downstream filter j and the application is considered to have failed.

From the expressions for \( I_j(C) \) and \( I_j(I) \) in Eq. (6) and Eq. (7), Eq. (1), Eq. (2) and Eq. (3) for filter j can be written as:

\[
p_j(C) = \frac{p_{j-1}(C)p_j(C | C) + p_{j-1}(I)p_j(C | I)}{p_{j-1}(C) + p_{j-1}(I)}
\]

\[
p_j(I) = \frac{p_{j-1}(C)p_j(I | C) + p_{j-1}(I)p_j(I | I)}{p_{j-1}(C) + p_{j-1}(I)}
\]

\[
p_j(N) = \frac{p_{j-1}(C)p_j(N | C) + p_{j-1}(I)p_j(N | I)}{p_{j-1}(C) + p_{j-1}(I)}
\]

Eq. (8), Eq. (9) and Eq. (10) hold for filters 2 through n. For the first filter, assuming the external input is always correct, \( p_{f1}(C) \) is given by \( p_{f1}(C) \) and \( p_{f1}(I) \) is given by \( p_{f1}(I) \). The application reliability is given by:

\[
R = \prod_{i=1}^{a-1} (p_{f1}(C) + p_{f1}(I)) \prod_{j=1}^{n-1} p_{j+1}(C)
\]

Eq. (11) indicates that for filters \( I \) through \( n-I \), the possibilities of both correct and incorrect outputs are regarded as success. For filter \( n \), however, only the scenario of correct output is regarded as success and the scenarios of incorrect and no output are regarded as failure.

Case 3: Topology with feedback for quality improvement, conservative policy.

To obtain an expression for application reliability in this case, first an expression for the average reliability for multiple processing iterations by filters \( n_1 \) through \( n_2 \) in the feedback loop is obtained. In the first round of processing, the probability that filters \( n_1 \) through \( n_2 \) process an element of input data without incurring a failure is given by

\[
\prod_{i=n_1}^{n_2} p_j(C) \prod_{i=n_1}^{n_2-1} p_j(C) \prod_{i=n_1}^{n_2} p_j(C)
\]

With probability \( q_1 \), the output produced by the cascade of filters \( n_1 \) through \( n_2 \) is of acceptable quality. Thus, with probability \( (1-q_1) \), the output produced is of unacceptable quality requiring a second round of processing. In the second round of processing, the probability that processing the cascade of filters \( n_1 \) through \( n_2 \) does not encounter a failure is given by

\[
\prod_{i=n_1}^{n_2} p_j(C) \prod_{i=n_1}^{n_2-1} p_j(C)
\]

With probability \( q_2 \), this output is of acceptable quality requiring no further action, and with probability \( (1-q_2) \) a third round of processing is necessary. Using the theorem of total probability [27], the probability that an output of acceptable quality is obtained at the \( k \)th iteration is given by

\[
\prod_{i=s+1}^{m} (1-q_i)q_i
\]

for \( 2 \leq s < m \).

The reliability of the cascade for the \( k \)th iteration is given by

\[
W_p = \prod_{i=j}^{n} p_{j,i+1} p_{j,i+1} \cdot W_p
\]

The extreme case, namely, \( I = m \) merits special consideration. In this case, the probability of obtaining acceptable quality output is \( (1-q_1)(1-q_2) \ldots (1-q_{m-1}) \).

Further, \( q_m = I \), since \( m \) is the maximum number of iterations that are possible with different filter configurations. In other words, if an output of acceptable quality cannot be obtained after \( m-I \) iterations through the cascade of filters \( n_1 \) through \( n_2 \), then the output produced in the \( m \)th iteration is considered acceptable as long as no failure occurs. Then the average reliability of processing by the cascade of filters \( n_1 \) through \( n_2 \), denoted \( W_j \) is given by Eq. (12).

\[
W_j = W(C)W_p + \sum_{i=1}^{m-1} \left( \prod_{i=1}^{m-1} (1-q_i)q_i \right) W(C) + \sum_{i=1}^{m-1} \frac{1}{W_p}
\]

The application reliability is given by:

\[
R = \prod_{i=1}^{n-1} (p(C)*p_{i+1}) + \prod_{i=n_2}^{n} p(C)*p_{i+1}
\]

Since the conservative policy is employed, \( p_j(C) \) is the same as \( p_j(C) \) for all the filters.

Case 4: Topology with feedback for reliability improvement, conservative policy.

Similar to the previous case, to obtain the application reliability first an expression for the expected reliability for multiple processing iterations by filters \( n_1 \) through \( n_2 \) is obtained using the following reasoning. If a failure occurs after the first round of processing, then a second round of processing is necessary. If the second round of processing also results in a failure, then a third round of processing becomes necessary. This can continue until all \( m \) iterations are exhausted. The probability that each iteration produces a correct output is given by \( W(C)W_p \) where

\[
W(C) = \prod_{i=n_1}^{n_2} p(C) \quad \text{and} \quad W_p = \prod_{i=n_1}^{n_2} p(C) \cdot W_p \]

Thus the probability that an iteration does not produce correct output, thereby requiring the next iteration is given by \( (1-W(C)W_p) \). The average reliability of processing in the feedback loop is given by:

\[
W_4 = \sum_{i=1}^{m} W(C)W_p(1-W(C)W_p)^{i-1}
\]
The application reliability is given by:

\[ R = \left( \prod_{i=1}^{a-1} p_i(C) p_{i+1} + \prod_{i=a}^{a} p_i(C) p_{i-1} \right) W_a \prod_{i=a+1}^{a} p_i(C) p_{i-2} \]  (15)

Since the conservative policy is employed, \( p_i(C) \) is the same as \( p_i(C) \) for all the filters.

4. Case study: Document Analysis and Understanding Application

This section illustrates the potential of the reliability analysis methodology using an industrial case study of a Document Analysis and Understanding Application.

Document understanding refers to the process of converting paper material such as books, magazines, journals, etc. into a searchable electronic form with information that is meaningful to both human beings and machines [31]. The conversion process may be referred to as "remastering". Typically, the input is document pages in a raster format (TIFF or BMP) and the output is searchable Portable Document Format (PDF) or eXtensible Markup Language (XML) documents. Document understanding systems remaster documents for use in the digital libraries of Web communities, such as the cognitive science community [1].

Currently, there are several commercial applications that provide various document understanding services, including WISDOM++,e [6] [7], Abby Fine Reader [2], and Adobe Acrobat Capture [5]. Most existing solutions are considered applications or programs, rather than systems, which are useful for end users but not for high volume data processing as they require extensive human intervention. To process a massive amount of data in a reasonable amount of time, an automated system that runs continuously, with minimal human intervention, and operates on multiple documents simultaneously is needed.

From a software perspective, the architecture is based on component-based software engineering principles. Components perform the document understanding functions, such as Optical Character Recognition (OCR), layout analysis, and logical structure analysis. Each component is self-contained and fairly independent, and provides well-defined services and functions. Each component is wrapped such that the wrapper handles inputs, outputs, and errors and provides the execution context. The architecture is built using a basic software framework principle: "Don’t call us, we will call you". Components have no explicit knowledge of each other, whether they are on the same worker machine or not. Processes run in parallel over a distributed workstation cluster. The system is composed of multiple components and scripts that run in parallel with monitoring components, watchdogs, loggers, etc. For simplicity, the most critical portion of the application is extracted to illustrate the reliability analysis methodology. These are the pipes and filters responsible for the analysis of documents. Fig. 8 depicts a simplified diagram of the pipeline that is used for the illustration of the reliability analysis methodology.

Note that for this analysis, pipes are control pipes; that is, they carry a description of the data to be processed but not the data itself. For this type of applications, the data itself is large (in Megabytes) and hence a storage server is used to store the data. The location of the data to be processed is carried in the task description which goes into the pipe.

Fig. 9 depicts the same architecture when a feedback loop is added. In this simplified view of the architecture, the most critical component is the Content Analyzer/Converter filter which performs a bulk of the analysis. To improve the reliability and the quality of the Converter filter, a feedback loop is added. The feedback loop works as follows: the output from the PDF verifier is used to determine whether it has reached acceptable quality or not. If yes, the output is fed to the Writer filter. If not, the feedback loop is used to reinvoke the Converter filter to redo the analysis. The analysis is performed in every feedback turn with different parameters and settings. For this system, a set of five different types of analyzers are used; some of which use the same technology with different parameter values (for example, changing the exposure in the input page) and others use different analysis technology (for example using polygonal or quadratic region analysis).

The conditional probabilities of the filters are shown in Table 3. These conditional probabilities could be obtained from expert opinion or domain knowledge. They could also be estimated by statistical testing of the components with test cases drawn from the operational profile [23]. The rationale behind this initial choice of these probabilities is as follows. The Tape Loader uploads the data from the archive tapes into the system. Only one type of errors were observed in this component, the component may not write the data correctly to the system disks, thus the value of \( p(N|C) \) is set to a small non zero value. It will never produce correct output, given incorrect input, hence \( p(C|I) \) is set to zero. The Writer filter component has proven to be sufficiently reliable so we set its \( p(C|C) \) to one. The filter reads output data and groups it into manageable, writeable disks. The Input TIFF filter reads the input and checks for its quality. It may decide that the input is correct while it is not and hence \( p(C|I) \) is set to this likelihood value. Also, there is a likelihood that it will produce incorrect output given that the input was correct, however, the second case is less likely and most of the errors observed from this filter belong to the first category. Thus, \( p(I|C) \) is set to a much lower value than \( p(C|I) \). The PDF Verifier is another stable filter that is based on a stable commercial component. This component was carefully selected to be highly reliable since based on its output a decision is made whether the quality of the output is acceptable or not. The Content Analyzer is the most critical or error prone component. It is a complex component that encompasses sophisticated image processing and analysis algorithms. It is the most important component in the system, at the same time it is the one that is most likely to fail. Therefore, a feedback loop is used to cover this component to decrease the likelihood of failures and to increase the possibility of producing output of
acceptable quality. For the feedback loop, the values of \( q_1 \), \( q_2 \), \( q_3 \), \( q_4 \), and \( q_5 \) are set to 0.96, 0.97, 0.98, 0.99, and 1.0 respectively. Since the number of iterations is five, \( q_5 \) is set to 1.0. The value of \( d \) is set to 1.0, and the value of \( m \) is set to 5.

Table 3: Conditional probabilities of filters

<table>
<thead>
<tr>
<th>Probability</th>
<th>TL</th>
<th>IT</th>
<th>CA</th>
<th>PDF</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p(C</td>
<td>C) )</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>0.98</td>
</tr>
<tr>
<td>( p(I</td>
<td>C) )</td>
<td>0.00</td>
<td>0.025</td>
<td>0.03</td>
<td>0.005</td>
</tr>
<tr>
<td>( p(N</td>
<td>C) )</td>
<td>0.05</td>
<td>0.025</td>
<td>0.02</td>
<td>0.015</td>
</tr>
<tr>
<td>( p(C</td>
<td>I) )</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.00</td>
</tr>
<tr>
<td>( p(I</td>
<td>I) )</td>
<td>0.00</td>
<td>0.492</td>
<td>0.492</td>
<td>1.00</td>
</tr>
<tr>
<td>( p(N</td>
<td>I) )</td>
<td>0.00</td>
<td>0.008</td>
<td>0.008</td>
<td>0.00</td>
</tr>
</tbody>
</table>

With these values of the parameters, the application reliability of the four variants identified in Section 3.2 was computed using the reliability expressions derived in Section 3.4 and these values are summarized in Table 4. The results in Table 4 indicate that for the topology without feedback, the opportunistic policy provides an improvement in the reliability over the conservative policy. For the topology with feedback, when the feedback loop is incorporated to improve quality, a significant degradation in the reliability does not occur due to the multiple processing iterations. In fact, the application reliability with feedback is close to the reliability in the conservative policy without feedback. Finally, the application reliability when the feedback loop is incorporated for reliability improvement is the highest among all the four cases. Intuitively this is expected, however, the expressions in this paper allow us to quantitatively assess the reliability improvement/degradation of each strategy.

Table 4: Summary of reliability estimates

<table>
<thead>
<tr>
<th>Case</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without feedback, conservative</td>
<td>0.8402</td>
</tr>
<tr>
<td>Without feedback, opportunistic</td>
<td>0.8519</td>
</tr>
<tr>
<td>With feedback (quality improvement)</td>
<td>0.8390</td>
</tr>
<tr>
<td>With feedback (reliability improvement)</td>
<td>0.9020</td>
</tr>
</tbody>
</table>

The reliability estimates reported in Table 4 are consistent with the expectations of the experts. Since the failure data from the actual application were never collected, a quantitative validation of these estimates could not be conducted. However, the important use of these analytical expressions is at design time, where the focus is on assessing the sensitivity of the application reliability to the filter/pipe reliabilities rather than obtaining a very accurate reliability estimate. Sensitivity analysis can be used to guide resource allocation to improve the application reliability effectively. Since the reliability estimates obtained from the methodology matched the designer’s expectations, it can enhance confidence in its use for such design-time analysis.

Next the potential of the reliability expressions for sensitivity analysis through several experiments is illustrated.

**Experiment 1:** The first experiment analyzes the sensitivity of the application reliability to \( p(C|C) \)s of the filters for linear topology without feedback employing both the conservative and opportunistic policies. Towards this end, \( p(C|C) \) of each filter was varied one at a time, in the range of 0.60 to 1.00 in steps of 0.05, while holding \( p(C|C) + p(I|C) = 1.0 \) and setting \( p(N|C) = 0.0 \). When \( p(C|C) \) of a given filter was varied, the remaining parameters of that filter and of all the other filters were held at the values reported in Table 3. Figs. 10, 11, 12, 13, and 14 show the variation in the application reliability for variations in \( p(C|C) \) for the Tape Loaders, Input Tiff, Content Analyzer, PDF Verifier, and Writer filters respectively. As expected, all the figures indicate that the opportunistic policy provides better reliability than the conservative policy. However, for lower values of \( p(C|C) \), the difference in the reliability between the opportunistic and conservative policies is higher for the Tape Loaders and Input Tiff filters. For the Content Analyzer, PDF Verifier, and Writer filters, the difference in the reliability between the opportunistic and conservative policies is uniform over the entire range of variation of \( p(C|C) \). Referring to the topology of pipes and filters in Fig. 8, it can be seen that the Tape Loaders and Input Tiff filters occur before the remaining three filters. As a result, the higher probability of incorrect processing of an input data element by these two filters can be partly compensated by the downstream filters in the opportunistic policy. This indicates that if limited resources are available, then it may be more desirable to enhance the reliability of the downstream filters and to provide them with an ability to mask the errors committed by the upstream filters in order to improve the application reliability. The expressions in this paper can facilitate such tradeoffs.

![Fig. 10: Reliability vs. \( p(C|C) \) of Tape Loaders filter (wo feedback, conservative and opportunistic policies).](image)

![Fig. 11: Reliability vs. \( p(C|C) \) of Input Tiff filter (wo feedback, conservative and opportunistic policies).](image)
Experiment 2: Fig. 12 indicates that the opportunistic policy does not provide a significant improvement in the application reliability over the conservative policy, even when $p(C|C)$ of the Content Analyzer filter is low. A similar observation can be made from Fig. 13 for the PDF Verifier filter. Another way to improve the application reliability is to employ a feedback loop which covers the Content Analyzer and PDF Verifier filters. The second experiment analyzes the sensitivity of the application reliability to $p(C|C)$ of the Content Analyzer and PDF Verifier filters in the presence of a feedback loop. $p(C|C)$ was varied over the same range using the same step size as in Experiment 1 for each of the Content Analyzer and PDF Verifier filters one at a time.

Experiment 3: The results of experiment 2 indicate that the feedback loop offers a significant improvement in reliability, especially when $p(C|C)$s of the Content Analyzer and PDF Verifier filters are low. This improvement in reliability, however, is achieved by sacrificing performance, since when filters in the loop are involved in multiple processing iterations for a single data element they are unavailable to process the subsequent or following elements of data. This backlog may increase the time taken to process subsequent elements of data. It may also propagate upstream, ultimately to the first filter, which may result in a loss of input data.

To prevent this, for given values of $p(C|C)$ for Content Analyzer and PDF Verifier filters, it is necessary to determine the threshold number of iterations through the feedback loop beyond which additional rounds of processing yield a negligible improvement in the application reliability. Through this experiment, the capability of the analytical expressions derived in this paper to determine the threshold number of iterations is illustrated. $p(C|C)$ for each (Content Analyzer and PDF Verifier) filter was varied one at a time, while holding the parameters of the others at their original values in Table 3. The range of variation of $p(C|C)$ was from 0.60 to 1.00 in steps of 0.1. For each value of $p(C|C)$, the maximum number of iterations $m$ was varied from 2 through 8 in steps of 1. Figs. 17 and 18 depict the application reliability as a function of $m$ for each value of $p(C|C)$ for the
Content Analyzer and PDF Verifier filters respectively. As expected, the figures suggest that as \( p(C|C) \) increases the threshold number of iterations beyond which a tangible improvement in reliability is feasible reduces.

![Fig. 16: Reliability vs. \( p(C|C) \) of PDF Verifier filter (with and without feedback)](image1)

![Fig. 17: Reliability vs. number of feedback iterations (Content Analyzer filter)](image2)

![Fig. 18: Reliability vs. number of feedback iterations (PDF Verifier filter)](image3)

**Experiment 4:** In the last experiment the sensitivity of the application reliability to the Content Analyzer and PDF Verifier filters when the feedback loop is employed to improve quality was assessed. \( p(C|C) \) for each one of these filters was varied in the same range and using the same step size as in Experiment 1. Fig. 19 depicts the application reliability as a function of \( p(C|C) \) for Content Analyzer and PDF Verifier filters. As expected, low values of \( p(C|C) \) result in low application reliability for the following reason. Referring to Figs. 4 through 7, the rate at which iteration-dependent probability of correct processing increases is tied to the initial value of \( p(C|C) \). Thus, when the initial value of \( p(C|C) \) is low, the rate of increase is also slow. The chance of failure in each iteration of the feedback loop remains significant, thereby reducing the application reliability. Thus, a tradeoff exists between the number of iterations that should be employed for quality improvement versus the application reliability, which will be driven by the initial values of \( p(C|C) \) for the filters involved in the feedback loop.

![Fig. 19: Reliability vs. \( p(C|C) \) of Content Analyzer and PDF Verifier filters (with feedback for quality)](image4)

**5. Related research**

Architecture-based software reliability analysis has been the focus of several research efforts in the past few years. Prevalent architecture-based techniques can be classified into three categories, namely, path-based [17] [32], state-based [14] [15] [19] and additive [30] as proposed by Goseva et al. [15]. In the path-based approach, reliability is estimated by considering the possible execution paths through the application architecture. As a result, the path-based approach provides only an approximate estimate of the reliability for an application which has infinite paths due to the presence of loops. Ammar et al. [4] evaluate software architecture specifications for error propagation using the path-based approach, but do not connect error propagation probabilities to the application reliability. Popic et al. [24] introduce error propagation and quantify its impact on application reliability, but the problem of path-based approaches still persists in this method. In the state-based approach, a control flow graph is used to represent the application architecture, which is then mapped to a state space model such as Discrete Time Markov Chain (DTMC) [10] [13] [25] [26], a continuous time Markov Chain (CTMC) [18], or a semi Markov process [21]. An estimate of the application reliability is obtained analytically from the solution of these models. The state-based approach can account for the presence of infinite paths arising due to loops using analytical methods. Additive models simply add the failure rates of the components to
obtain the failure rate of the application [30]. These models do not consider software architecture explicitly.

Chen et al. [28] present a methodology to map the constraints of a few simple, well-behaved architecture styles to state space models. Abd-Allah [3] identify the issues involved in using reliability block diagrams to analyze the reliability of different architecture styles.

Prevalent state-based techniques cannot consider the impact of error propagation and possible downstream error correction on the application reliability. Second, the state space models cannot account for a deterministic number of feedback iterations, with component reliabilities dependent on the number of iterations. The reliability analysis methodology proposed in this paper considers the impact of these two factors which may be employed to improve application reliability. Hence the reliability estimate produced by the methodology may be more accurate than the prevalent techniques.

6. Conclusions and future research

This paper presented a reliability analysis methodology for the pipe and filter architecture style. Two variants of the topological organization of the pipes and filters were chosen, namely, linear topology without feedback and linear topology with feedback, where the feedback loop is employed either to improve quality or to improve reliability. The methodology considered the impact of: (i) error propagation and downstream error correction, and (ii) deterministic number of iterations through the feedback loop, with filter reliabilities dependent on the number of iterations, on the application reliability. The potential of the methodology to obtain a reliability estimate as well as to facilitate sensitivity analysis was illustrated using an industrial case study of a Document Analysis and Understanding Application.

The present paper considered only point estimates of the reliabilities of pipes and filters. Extending the methodology to propagate the variances in the reliability estimates of the pipes and filters to the variance in the application reliability estimate is a topic of future research. Development of reliability analysis methodologies for other architecture styles such as the event-driven and database (repository) styles is also a concern of the future.

Acknowledgments

This research is supported by EpSCoR and Workforce Development grants from Connecticut Space Grant Consortium, Large Grant from the University of Connecticut Research Foundation and the following grants from the US National Science Foundation (CNS-0406376 and CNS-SMA-0509271).

References


Author Bios

Swapna S. Gokhale is an Assistant Professor in the Dept. of Computer Science and Engineering at the University of Connecticut. She received her M.S. and Ph.D. in Electrical and Computer Engineering from Duke University in 1996 and 1998 respectively and her B.E. (Hons.) in Electrical and Electronics Engineering and Computer Science from the Birla Institute of Technology and Science, Pilani, India, in 1994. Prior to joining UConn, she was a Post Graduate Researcher at the University of Califorina, Riverside and a Research Scientist in the Applied Research Division of Telcordia Technologies (formerly Bell Communications Research). Her research interests lie in the areas of software reliability, software performance, QoS assurance of wireless and wireline networks and application-level intrusion detection.

Sherif Yacoub is a member of research staff at HP Laboratories. He specializes in technologies for building reliable large-scale software systems, including design patterns, architecture, modeling, meta-modeling, frameworks and reliability. He has authored over 40 technical papers and has presented at TOOLS, UML and ISSRE. He has co-authored the textbook Reuse-based Software Engineering published by John Wiley and Sons in 2001.