A Reliability Model for Nonhomogeneous Redundant Software Versions with Correlated Failures

Walter J. Gutjahr
Department of Statistics and Decision Support Systems
University of Vienna, Austria

Abstract: The article develops a technique for computing the reliability (1) of a recovery block with fault–free test segment, and (2) of a k-of-n voting system, on the assumption that the failures of the versions are positively correlated, and that the single failure probabilities ("unreliabilities") may differ. In that, the suggested stochastic model generalizes a model previously proposed by Nicola and Goyal assuming equal single unreliabilities. Numerical computation procedures implementing the presented approach are outlined. The technique is evaluated on empirical data from a large-scale experiment by Eckhardt et al. It turns out that the obtained predictions are distinctly better than those based on the assumption of equal version unreliabilities or of independent failures.

Keywords: Correlated failures, error recovery, fault–tolerance, multi–version programming, software reliability.

1 Introduction

An important issue in the evaluation and optimization of fault–tolerant software systems is the adequate modeling of the effect of *software redundancy.* In its simplest form, a program using redundancy contains k versions with the same functionality, and in the simplest modeling approach, the program is assumed to work correctly if at least one of the versions contained in it produces the correct output — an assumption that reflects a *recovery block* structure with fault–free test segment: A recovery block can be described in pseudo-code as

```
ensure acceptance test
  by primary version (#1)
  else by alternate version (#2)
  ...
  else by alternate version (#k)
else error
```

(see [16]), and in many cases, the acceptance test is much easier to perform than the computation itself, such that its failure probability can be considered as negligible, compared to that of the versions.

An alternative to the use of a recovery block is that of a k-of-n *voting system*, typically applying a majority voting rule, such that the system works correctly if and only if at least

$k = \lceil (n+1)/2 \rceil$ of the n implemented versions produce correct output. For both types of failure-tolerant structures, techniques for reliability prediction are required.

It has been recognized (see Knight and Leveson [11]) that redundant software versions cannot be supposed to fail independently of each other: usually, there is a rather strong positive correlation between their failure behavior. Models for the overall program reliability that take this correlation into account have been developed (see Eckhardt and Lee [5], Nicola and Goyal [13], Tomek, Muppala and Trivedi [16]) and used for optimization purposes (see [8]).

A drawback of the mentioned models is the fact that they are based on the assumption of *identical* single reliabilities of the redundant versions. This assumption needs not to be satisfied in applications. For example, versions may have been written by developer teams with different skill and experience, or in different programming languages, or they may have been exposed to a different amount of testing or of previous trial in the application field. All these factors may lead to differing reliability estimates for the single versions. Our aim is therefore to generalize the modeling technique of [13] (which is a specific version of the approach in [5]) to the *nonhomogeneous* case, i.e., the case where the reliabilities of the redundant versions may vary. We try to do this under certain constraints:

- The extended model should contain the Nicola–Goyal model [13] as the special case of homogeneity.
- It should still be possible to interpret the extended model in the context of the concepts of [5], although it generalizes them as far as version non-homogeneity is concerned.
- On the assumptions of the model, the overall program reliability should be uniquely defined, and its numerical computation should be feasible.

It will be shown that it is possible to satisfy all these three constraints by the model suggested below.

2 Related Work

Correlated failures combined with *different* version unreliabilities are already implicit in models as developed by Saglietti and Ehrenberger [14], Saglietti [15] or Arlat et al. [1]. However, these models typically involve only *two* redundant versions, and it is not clear how they could be generalized to a larger number of versions. For two versions, three parameters, namely p_i $(i = 1, 2)$, giving the two failure probabilities, and p_{12}, giving the joint failure probability, are sufficient for the specification of the model (cf. [14]). For a larger number of versions, the situation gets soon much more complex.

The pioneering work by Eckhardt and Lee [5], the ideas of which will be shortly outlined at the beginning of Section 3, allows the treatment of an arbitrary number of redundant versions, but at the price of the assumption that all versions have the same failure probabilities. This feature is also inherited by the Nicola-Goyal approach [13] which takes the framework of [5] and specifies it by assuming a particular type (Beta-binomial) of the failure intensity distribution (for details, see Section 3).

Tomek et al. [16] start with a recapitulation of the model by Eckhardt and Lee on the specific assumption of [13], i.e., a Beta-binomial failure intensity distribution with single-version failure probabilities and correlation level as parameters, but they also propose an alternative within the Eckhardt-Lee framework: to approximate the failure intensity distribution by using the pairwise failure correlation between modules. Moreover, they consider two types of failure correlation other than that between different versions of the same program: (a) correlations between acceptance test executions, and (b) clustering in the input data stream: if a certain input is "difficult", then there is an increased probability that the next input(s) will also be difficult. *Stochastic Reward Nets* (SRNs) are applied then as tools for modeling all mentioned types of dependencies and for obtaining numerical results.

Also Bondavalli et al. [4] explicitly study the input clustering effect: They assume particular transition probabilities between iterations with success, with benign failures and with catastrophic failures. The model is evaluated for three failure-tolerant structures, namely recovery blocks, multiple version programming (i.e., voting systems), and Self-Configuring Optimal Programming. Their model assumptions are well-suited for fault-tolerant software executed iteratively (e.g., process control applications or safety systems with periodic scheduling). In the present paper, we are not concerned with the input clustering effect, since the focus is here rather on single program executions instead of the processing of the entire input stream as it is important for the investigation of real-time systems. In particular, we do not consider typical problems of fault-tolerant *hard real-time systems*, as they have been addressed, e.g., by Kopetz [10]. Instead, we concentrate on removing the main limitation of the Eckhardt-Lee framework, the assumption that unreliabilities of redundant versions are equal.

3 The Model

3.1 Recovery Blocks with Fault-Free Test Segments

Let D be the set of possible inputs to the given program. The fundamental idea of Eckhardt's and Lee's framework [5] is to suppose that on a *fixed* input $x \in D$, the redundant versions $1, \ldots, k$ fail *independently* from each other (which is to be expected if they have been developed independently); the correlation of the failures of the versions on a *random* input $X \in D$ during the usage of the program results then from the fact that the failure probabilities in x are not the same for all $x \in D$. Let $p(x)$ denote the fraction of programs, out of all programs potentially written to the given specification, that fail on input x, i.e., the probability that a randomly chosen program out of this population fails on x. If also the input is chosen randomly from D, say, according to the operational distribution in the application field, then also $P = p(X)$ gets a random variable with a certain distribution $\mathcal{D}(P)$, which is called the *failure intensity distribution*. (In Section 4, we shall use the term "failure intensity distribution" also when we do not refer to the fraction of failing programs within the basic population, but within the set of redundant program versions in the fault-tolerant system. From the context, it will be clear which interpretation is intended.)

Nicola and Goyal suggest to model $\mathcal{D}(P)$ by a *Beta distribution* Beta (α, β) with parameters α and β chosen as

$$\alpha = \frac{\pi}{\theta}, \qquad \beta = \frac{1-\pi}{\theta}, \tag{1}$$

where π is the (expected) unreliability of a fixed version (i.e., the expected value of the random variable P), and $\theta \geq 0$ is a so–called *correlation level*: The limiting case $\theta = 0$ is the case where all versions fail independently of each other on a random input (no correlation), whereas the limiting case $\theta = \infty$ is the case where the failure behavior of all versions is identical (maximal correlation). The density of the Beta distribution is given by

$$f(p) = \frac{p^{\alpha-1}\,(1-p)^{\beta-1}}{B(\alpha,\beta)}, \tag{2}$$

where

$$B(\alpha,\beta) = \int_0^1 u^{\alpha-1}\,(1-u)^{\beta-1}\,du \tag{3}$$

is the Beta function. Whenever we shall prefer to express the Beta function rather in the parameters π and θ than in the parameters α and β, we shall use the notation Beta $[\pi,\theta]$ instead of Beta $(\pi/\theta,\ (1-\pi)/\theta)$.

The technique presented in this paper relies on the assumption that there is some *basic* function $p_0(x)$ reflecting the hypothetical fraction of programs failing on x in the overall population of programs potentially written to the given specification, and that, to each version i, there is a *specific* function $p_i(x)$ as well, reflecting the fraction of programs failing on x in the sub–population of programs out of which version i is supposed to have been selected ($i = 1,\ldots,k$). Moreover, we assume that $p_0(x)$ and $p_i(x)$ ($i = 1,\ldots,k$) are linked in a certain way, ensuring that, as in [13], the random variables $P_i = p_i(X)$ have Beta distributions. It will be shown that the choice of $p_0(x)$ is arbitrary; $P_0 = p_0(X)$ needs not even to be Beta–distributed. The only required assumption on P_0 is the technical constraint that the distribution function of P_0,

$$F_0(p) = Pr\{P_0 \leq p\} = \int_D 1_{\{p_0(x)\leq p\}}\,dQ(x), \tag{4}$$

where 1_A denotes the indicator function of set A and Q denotes the operational distribution on D, is differentiable on $[0,1]$, such that there exists a density function $f_0(p) = F_0'(p)$. In the integral notation above, we assume that D is either a finite set, a countably infinite set, or a (measurable) subset of $\mathbb{R}^n$; the Lebesgue integral works in all these cases.

It is supposed that the (expected) unreliability π_i of each single version i ($i = 1,\ldots,k$) can be estimated. Furthermore, we suppose that there is some common correlation level $\theta \geq 0$ for the k versions. We want to achieve that the distribution $\mathcal{D}(P_i)$ of random variable $P_i = p_i(X)$ is a Beta $[\pi_i,\theta]$ distribution. This can be done as follows:

Let F_i be the distribution function of a Beta $[\pi_i,\theta]$ distribution. For $\theta > 0$, F_i is strictly monotone and can hence be inverted. We define

$$\varphi_i = F_i^{-1} \circ F_0 \qquad (i = 1,\ldots,k) \tag{5}$$

and assume that the link between $p_0(x)$ and $p_i(x)$ is given by

$$p_i(x) = \varphi_i(p_0(x)) \qquad (i = 1,\ldots,k). \tag{6}$$

If $F_i(x)$ jumps at $x = u$ from some value $p' < p$ to some value $p'' > p$, such that $F_i^{-1}(p)$ is not defined (this happens in the extreme case $\theta = 0$), we simply set

$$F_i^{-1}(p) = F_i^{-1}(p'') = u$$

in the definition of φ_i.

Example.

The following simple example is intended to provide the reader with an intuitive access to the described model assumptions. Imagine an overall population of programs with average failure probability of $\pi_0 = 0.1$ and a subpopulation of programs for which the average failure probability is $\pi_1 = 0.2$, and suppose that the correlation level is $\theta = 0.4$. Then the Nicola-Goyal model assumes the two (cumulative) failure intensity distribution functions $F_0(p)$ (solid curve) and $F_1(p)$ (dashed curve) depicted in Fig. 1.

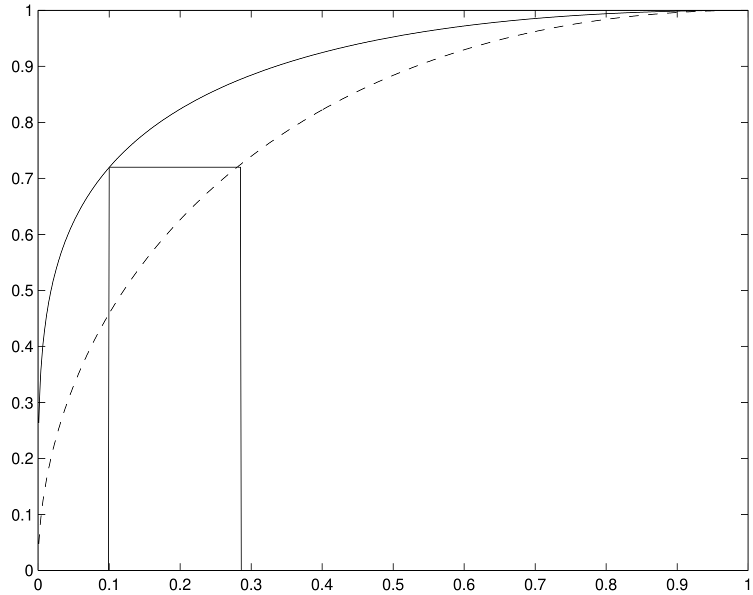

Fig. 1. Cumulative failure intensity distribution function F_0 (solid) and F_1 (dashed) in the Example.

It can be observed that for the overall population, the model predicts the probability of an input for which a fraction of at most, say, 10% of programs fail, as $F_0(0.1) = 0.72$, whereas for the (less reliable) subpopulation, the probability of such an input is predicted as $F_1(0.1) = 0.45$. In order to get again a probability equal to 0.72 of an input leading to not more than a certain threshold fraction of failing programs in the *subpopulation*, the threshold must be increased from 0.1 to 0.28. The value 0.28 is here computed as $0.28 = F_1^{-1}(F_0(0.1))$ (Fig. 1 illustrates this construction). The simplest model assumption explaining equal values of the respective cumulative distribution functions F_0 and F_1 in the two populations at the two considered points is that each input x with a fraction $p_0(x)$ of failing programs in the overall population has a fraction $p_1(x) = F_1^{-1}(F_0(p_0(x))$ of failing programs in the subpopulation: the set of all x for which $p_0(x) \leq 0.1$ is then the same set as the set of all x for which $p_1(x) \leq 0.28$. This consideration motivates eqs. (5) – (6).

Proposition 3.1. $p_i(X)$ is Beta $[\pi_i, \theta]$–distributed.

Proof.

$$Pr\{p_i(X) \leq u\} = Pr\{\varphi_i(p_0(X)) \leq u\} = Pr\{p_0(X) \leq \varphi_i^{-1}(u)\}$$
$$= F_0(\varphi_i^{-1}(u)) = (F_0 \circ F_0^{-1} \circ F_i)(u) = F_i(u).$$

□

The quantity of interest in our context is the unreliability of the program, i.e., the probability that all of the k given versions fail on a random input X. In accordance with the notation in [13], we denote this quantity by $b_k(k)$. Our main result is the following:

Proposition 3.2. The probability $b_k(k)$ that, under the assumption (6), all k versions fail on a random input, is given by

$$b_k(k) = \int_0^1 \left(\prod_{i=1}^{k} \varphi_i(p) \right) f_0(p)\, dp \tag{7}$$

with $\varphi_i = F_i^{-1} \circ F_0$, $f_0 = F_0'$. This value is *independent* of the chosen basic distribution F_0.

Proof. See the Appendix.

The invariance property stated in Proposition 3.2 allows a variety of possible choices for the basic distribution F_0, all leading to the same value for $b_k(k)$. The decision for a particular F_0 is only a question of how to do the numerical computation as easily as possible. Let us outline two special choices:

Choice (a):

$F_0(p) = p$, i.e., $f_0(p) = 1$ for all p. Then (7) reduces to

$$b_k(k) = \int_0^1 \prod_{1=1}^{k} \varphi_i(p)\, dp,$$

where

$$\varphi_i(p) = F_i^{-1}(F_0(p)) = F_i^{-1}(p).$$

This yields

$$b_k(k) = \int_0^1 \prod_{i=1}^{k} F_i^{-1}(p)\, dp. \tag{8}$$

Choice (b):

It is possible as well to choose F_0 as identical with one of the distributions F_i, say, with F_1. Then (7) takes the form

$$b_k(k) = \int_0^1 \left(\prod_{i=1}^{k} \varphi_i(p) \right) f_1(p)\, dp, \tag{9}$$

where now $\varphi_i = F_i^{-1} \circ F_1$.

As stated before, we have taken care that our approach generalizes that in [13], i.e., that it contains the Nicola–Goyal model as the special case $\pi_1 = \ldots = \pi_k$. It is easy to see that this is true: If the values π_i are identical, then also $F_1 = \ldots = F_k$. Then, however, the functions φ_i in (9) become equal to identity:

$$\varphi_i = F_i^{-1} \circ F_1 = F_1^{-1} \circ F_1 = id.$$

Hence, in this case,

$$b_k(k) = \int_0^1 p^k\, f_1(p)\, dp = \frac{1}{B(\alpha,\beta)} \int_0^1 p^k \cdot p^{\alpha-1}(1-p)^{\beta-1}\, dp = \frac{B(\alpha+k,\beta)}{B(\alpha,\beta)}$$

as in [13], eq. (2.2), or [8], eq. (4).

Also another special case is of interest, namely that of independence. If $\theta = 0$, the distribution Beta $[\pi_i, \theta]$ reduces to the point mass in π_i, i.e., $F_i(p) = 0$ if $p < \pi_i$, and $F_i(p) = 1$ otherwise. Then, using choice (a) above for F_0, we get

$$\varphi_i(p) = F_i^{-1}(F_0(p)) = F_i^{-1}(p) = \pi_i$$

(cf. the remark after eq. (6)). Therefore,

$$b_k(k) = \int_0^1 \left(\prod_{i=1}^k \pi_i\right) f_0(p)\, dp \;=\; \prod_{i=1}^k \pi_i$$

in accordance with the product rule for probabilities of independent events.

The most tedious part in the numerical evaluation of (7) is the computation of the k values $\varphi_i(p) = F_i^{-1}(F_0(p))$ to each p, so let us consider this task in more detail. In the sequel, the index i will be omitted for the sake of simplicity. We have to determine the number x satisfying $x = F^{-1}(F_0(p))$, i.e., the solution of

$$F(x) - F_0(p) = 0. \tag{10}$$

Let

$$B(\alpha,\beta,x) = \int_0^x u^{\alpha-1}(1-u)^{\beta-1} du, \tag{11}$$

such that $B(\alpha,\beta) = B(\alpha,\beta,1)$ (cf. (3)). Then by (2),

$$F(x) = \int_0^x f(u) du = \frac{B(\alpha,\beta,x)}{B(\alpha,\beta)}. \tag{12}$$

For the computation of the root of (10), a Taylor expansion of $B(\alpha,\beta,x)$ is useful: By the binomial theorem, it is not difficult to derive that

$$B(\alpha,\beta,x) = \sum_{j=0}^{\infty} \frac{(-1)^j}{\alpha+j} \binom{\beta-1}{j} x^{\alpha+j}. \tag{13}$$

In [13], a typical value for θ, computed from an empirical study, is indicated as $\theta = 0.018....$ So we may assume that θ is usually smaller than unity. On the other hand, as argued in [8], Section 4, we may expect that in practical applications, π is small, such that $\beta = 1/\theta - \pi/\theta$ can be expected to be larger than unity in most applications. Since usually the precision of the estimate of θ (and hence also β) is not too large anyway, it is advisable to approximate β by an *integer*. In this case, the Taylor expansion (13) terminates after a certain number of steps, i.e., $B(\alpha, \beta, x)$ is a polynomial in x.

For parameter values α and β in the indicated range of practical interest, the following auxiliary result is applicable:

Proposition 3.3. For $\alpha \leq 1$, $\beta \geq 1$, and $0 \leq x \leq 1$, the function

$$\Psi(x) = F(x) - F_0(p)$$

on the l.h.s. of (10) is non-decreasing and concave.

The proof is straightforward and omitted for the sake of brevity.

Consequently, it is easy to compute the root of $\Psi(x)$ as follows:

- Start with $x = 0.1$ and divide x by 10 until $\Psi(x) < 0$.
- With the resulting value for x as the initial value, apply Newton's method.

Because of Proposition 3.1, Newton's method converges to the root. A sufficient accuracy can be obtained within very few iterations. Because of (11), the required function $\Psi'(x)$ is given by

$$\Psi'(x) = \frac{x^{\alpha-1}\,(1-x)^{\beta-1}}{B(\alpha,\beta)}.$$

All that remains to do in the final step is a numerical integration of the function

$$\left(\prod_{i=1}^{k} \varphi_i(p)\right) f_0(p)$$

in the interval $[0, 1]$.

In this way, we get the following numerical procedures for computing $b_k(k)$, based on the computation of $B(\alpha, \beta, x)$ by (13):

We choose the basic distribution F_0 as a Beta distribution. Let α_0 and β_0 be its parameters, and let $\alpha[\,]$ and $\beta[\,]$ be the arrays of the numbers α_i resp. β_i.

function `phi` $(\alpha, \beta, \alpha_0, \beta_0, p)$ {
  $c := B(\alpha_0, \beta_0, p) \cdot B(\alpha, \beta) \,/\, B(\alpha_0, \beta_0)$;
  $x := 0.1$;
  while $(B(\alpha, \beta, x) > c)$
    $x := x/10$;
  repeat until accuracy suffices
    $x := x - (B(\alpha, \beta, x) - c)\,/(x^{\alpha-1}\,(1-x)^{\beta-1})$;
  return x;
}

```
function integrand (k, α[], β[], α_0, β_0, p) {
  q := 1;
  for i := 1 to k
    q := q · phi (α[i], β[i], α_0, β_0, p);
  return q · p^(α_0 − 1) · (1 − p)^(β_0 − 1) / B(α_0, β_0);
}

function unreliability (k, α[], β[], α_0, β_0) {
  return result of numerical integration of integrand (k, α[], β[], α_0, β_0, p)
    in the interval p ∈ [0, 1], e.g., by Simpson's rule;
}
```

For the numerical quality of the indicated procedures, a suitable choice of the parameters α_0 and β_0 is important. The values where the integrand is essentially larger than zero should neither be concentrated near $p = 1$, as it is the case if $\alpha_0 = \beta_0 = 1$ is set (choice (a) in Section 2), nor near $p = 0$, as it is the case if α_0 and β_0 are chosen identical to some α_i resp. β_i (choice (b) in Section 2). For α_i and β_i in the range of practical interest, $\beta_0 \approx \theta^{-1/2}$ has turned out as a good choice. By an adaptive procedure, it is the easy to compute a value for α_0 leading to a balanced shape of the integrand.

It is immediately seen that the computational complexity of the solution algorithm is given by $O(\sigma \cdot k \cdot \nu \cdot t)$, where σ is the number of function evaluation points for doing the numerical integration, k is the number of versions, ν is the number of iterations in the application of Newton's method, and t is the number of terms in (13) used for the computation of $B(\alpha, \beta, x)$ (if, as suggested, β is rounded to an integer, then $t = \beta$). In our computations for the results of Section 4, the choice $\sigma = 100$ and $\nu = 10$ was sufficient for obtaining four significant decimal digits of precision. This choice led to a runtime of a MATLAB program under Windows 2000 on a PC Pentium 3 with 600 MHz and 132 MB RAM of less than 1 sec for $k = 2$ and only about 4 sec for $k = 10$.

Example. Let $k = 3$,

$$\pi_1 = 0.001, \quad \pi_2 = 0.0002, \quad \pi_3 = 0.0001$$

and $\theta = 0.02$, such that

$$\alpha_1 = 0.05, \quad \alpha_2 = 0.01, \quad \alpha_3 = 0.005,$$

and β_1, β_2 and β_3 are almost identical to 50. Our MATLAB implementation of the algorithm computes an unreliability of

$$b_3(3) = 1.844 \cdot 10^{-7}$$

for the system. To test the numerical quality, let us also consider the homogeneous case, e.g., by setting $\pi_1 = \pi_2 = \pi_3 = 0.0002$, and $\theta = 0.02$ as before. For this instance, our procedures compute the result

$$b_3(3) = 1.530 \cdot 10^{-7},$$

which is exactly the same value as obtained from eq. (2.2) in [13].

3.2 Voting Systems

Our approach can also be applied to the reliability estimation of k-of-n *voting systems*, since, as shown below, the knowledge of the coincident failure probability of each subsystem composed of k out of n given program versions allows the estimation of the complete failure intensity distribution of the total system of n versions. From the last distribution, one immediately derives the reliability of a k-of-n voting system, which is defined as the probability that at least k of the n versions produce correct output. (In the literature, also other types of voting systems resp. definitions of their reliability are considered; we follow here the definition in [13].)

Let us assume that versions with indices $1, \ldots, n$ are given, for which single failure probabilities $\pi_1, \ldots, \pi_n$ and a correlation level θ can be estimated. By means of eq. (7), the coincident failure probability $u(i_1, \ldots, i_k)$ of each subsystem consisting of versions $i_1, \ldots, i_k$ can be estimated, i.e., the probability that *all* these versions fail. By D, we denote again the input domain. Let $A \subseteq D$, and let $Pr(A)$ be the probability that the random input X (according to the operational distribution) is contained in A. Furthermore, let A_i denote the set of all inputs $x \in D$ on which version i fails, such that $u(i_1, \ldots, i_k)$ is an estimate of $Pr(A_{i_1} \cap \ldots \cap A_{i_k})$. Applying the well-known Inclusion-Exclusion Principle of elementary probability calculus, we get

$$\begin{aligned} q_s \;=\; & Pr\{\text{exactly } s \text{ versions fail on } X\} = \; Pr\{X \text{ is contained in exactly } s \text{ of the sets } A_i\} \\ & = \sum_{t=s}^{n} (-1)^{t-s} \binom{t}{s} \sum Pr(A_{i_1} \cap \ldots \cap A_{i_t}), \end{aligned} \tag{14}$$

where the second sum on the r.h.s. is over all subsets $\{i_1, \ldots, i_t\}$ of $\{1, \ldots, n\}$ with exactly t elements. An estimate of q_s is obtained by inserting $u(i_1, \ldots, i_t)$ for $Pr(A_{i_1} \cap \ldots \cap A_{i_t})$. The numbers q_s ($s = 0, \ldots, n$), however, just represent the failure intensity distribution of the system of n versions. The unreliability of the k-of-n voting system is then obviously given by

$$Pr\{\text{more than } n-k \text{ versions fail on } X\} \;=\; \sum_{s=n-k+1}^{n} q_s. \tag{15}$$

4 An Evaluation on Empirical Data

To check the validity and applicability of our approach, we have re-analyzed empirical data from a large-scale experiment by Eckhardt et al. [6]. In this experiment, 40 programmers produced 20 versions of a complex aero-space application (an inertial navigation system). The application can run under six different input conditions ($S_{0,0}$, $S_{0,1}$, $S_{1,0}$, $S_{1,1}$, $S_{2,0}$ and $S_{2,1}$), which may be considered as forming six non-overlapping subdomains of the input domain. In total, 920746 test cases were executed and evaluated for each of the 20 versions. The main numerical results of the experiment are the failure counts of each version (Table I in [6]) and the failure intensity distributions (Table II in [6]), both for each of the six subdomains. For the ease of the reader, we reproduce these two tables as Table I and Table II below.

Version number	$S_{0,0}$	$S_{0,1}$	$S_{1,0}$	$S_{1,1}$	$S_{2,0}$	$S_{2,1}$
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	2	4	51	9	63
5	0	1	0	0	0	0
6	1	0	0	0	0	0
7	0	3	0	12360	110	41032
8	196	561	14	128	7	89
9	187	279	0	0	0	0
10	0	2	21	80	10	67
11	14	179	20	155	13	41662
12	0	0	0	0	0	0
13	0	1	0	0	0	0
14	0	0	13	45	7	36
15	2	46	0	0	0	0
16	0	0	0	0	0	0
17	0	0	14	47	7	42
18	53	194	13	55	7	40
19	0	0	0	0	0	0
20	0	0	0	0	0	0
Total number of test cases	309442	134135	129999	101151	102510	143509

Table I. Eckhardt et al. [6] results on version failure counts for the six subdomains.

For each fixed subdomain, we may divide the failure count of each version i by the number of test cases for this subdomain, which yields the (observed) unreliability π_i of version i for the considered subdomain. E.g., for $S_{0,0}$ and version 8, the observed value of π_i is $196/309442 = 0.0006334$.

Proportion of versions failing together	$S_{0,0}$	$S_{0,1}$	$S_{1,0}$	$S_{1,1}$	$S_{2,0}$	$S_{2,1}$
0/20	0.999364	0.995707	0.999722	0.875038	0.998800	0.568626
1/20	0.000032	0.002214	0.000154	0.124040	0.001044	0.285362
2/20	0.000433	0.000626	0.000008	0.000386	0.000078	0.145684
3/20	0.000126	0.000171	0.000008	0.000089	0.000010	0.000028
4/20	0.000039	0.000939	0.000008	0.000020	0.000000	0.000028
5/20	0.000006	0.000343	0.000069	0.000109	0.000010	0.000084
6/20	0.000000	0.000000	0.000023	0.000189	0.000039	0.000111
7/20	0.000000	0.000000	0.000008	0.000119	0.000010	0.000049
8/20	0.000000	0.000000	0.000000	0.000010	0.000010	0.000028

Table II. Eckhardt et al. [6] results on proportions of versions failing together for the six subdomains. For the proportions 9/20 to 20/20, all values are zero (at most, 8 versions failed simultaneously).

Our aim is now to predict the failure intensity distributions in Table II from the version unreliabilities π_i, as obtained from Table I. We do this on three different model assumptions:

(1) Versions fail independently from each other; each version has the observed unreliability π_i.

(2) Failures are correlated, and for each version, the unreliability is the same. (This is the Nicola-Goyal model.) As an estimate of the version unreliability, we take the average $\bar{\pi}$ of the observed values π_i. In other words, $\bar{\pi}$ is taken as the expected unreliability of a randomly chosen version.

(3) Failures are correlated, and the unreliabilities of the versions need *not* to be identical. This is our model. The coincident failure probability of k versions is computed as described in Section 3. For the version unreliabilities, we take, as in model assumption (1), the observed values π_i.

In order to avoid trivial results, we selected, for each of the six input conditions $S_{0,0}$ to $S_{2,1}$, only those versions that have failed at least once during the experiment (the six versions 6, 8, 9, 11, 15 and 18 for $S_{0,0}$, ten versions for $S_{0,1}$, etc.), and composed them conceptually to recovery blocks or voting systems. In the following, we will call these versions the *unreliable* versions w.r.t. the given input condition for shortness.

Of course, since our computations are based on the observed unreliabilities π_i and the observed average unreliability $\bar{\pi}$, the predictions can be expected to be better than they would be if we would have used a-priori estimates $\tilde{\pi}_i$ resp. $\tilde{\bar{\pi}}$ of π_i resp. $\bar{\pi}$, as they are required in real applications. However, we are only interested in a comparison of the predictive capacity of the three different computational approaches. In such an investigation, the influence of the models themselves should be isolated from the additional, random effect of an inaccuracy in the estimation of π_i and $\bar{\pi}$. This can be achieved by doing the evaluation based on the real values.

Approach (2) and (3) require that, in addition, a correlation level θ is estimated. The real correlation level varies with parameters as program complexity, programming languages, degree of independence during development etc. Up to now, no rule of thumb, based on published experience, for the determination of θ seems available. So we decided to take for θ the weighted average of the observed correlation levels for $S_{0,0}$ to $S_{2,1}$ (the weights are the subdomain sizes), computed from Table II by eq. 2.6 of [13], which turned out as $\bar{\theta} = 0.40$. This procedure might seem questionable, since, as stated, it is just the content of Table II we want to predict. Nevertheless, the reader should notice that except of the single parameter value $\bar{\theta}$, we do not use any other information from Table II for the prediction, and that the comparison is fair insofar as $\bar{\theta} = 0.40$ is used in both models requiring information on the correlation level (Models 2 and 3). Moreover, the real values of the correlation level vary considerably between the six subdomains, and we only take the average, which is a very rough estimate for the real situation in a specific $S_{i,j}$.

To verify the predictions based on Model (1) – (3), we compare them with the following three types of information which can be obtained from Table II for each input condition:

(a) the relative failure frequency of a *recovery-block* with fault-free test segment, containing all unreliable versions w.r.t. the considered input condition (e.g., the relative frequency with which all the six versions 6, 8, 9, 11, 15 and 18 fail under $S_{0,0}$, which is actually zero by Table II);

(b) the relative failure frequency of a *majority voting system*, containing all unreliable versions w.r.t. the considered input condition (e.g., the relative frequency with which the 4-of-6 voting system with versions 6, 8, 9, 11, 15 and 18 fails under $S_{0,0}$);

(c) the failure intensity distribution given by Table II itself.

Of course, we have also the results for recovery blocks and voting systems composed of subsets of the sets of unreliable versions, but they cannot be verified by the data available from [6].

Table III shows the results of evaluation (a). Note that a relative failure frequency of 0.000001 in Table III corresponds approximately to one failure during test, since, as mentioned, about 920000 test cases have been evaluated. It can be observed that Model (1) predicts no failure during test for all six input conditions, which is clearly over-optimistic. In view of the famous study by Knight and Leveson [11] showing that the independence assumption for redundant versions is to be rejected at a statistical significance level of 0.99, this result is not surprising. Now let us consider the results of the models (2) and (3). For the subdomains $S_{1,0}$ and $S_{2,0}$ where the failure rates π_i of the versions are not varying too much (factor less than 20 between most and least reliable selected version), both models yield predictions of an acceptable order of magnitude. This is not true anymore for the subdomains $S_{0,0}$, $S_{0,1}$, $S_{1,1}$ and $S_{2,1}$ with a larger variance of failure rates: here, the prediction of Model (3) is still rather good, while that of Model (2) drastically overestimates the unreliability of the recovery block: Model (2) is over-pessimistic. Model (3), on the other hand, has no systematic bias towards "optimism" or "pessimism".

Table IV shows the results of evaluation (b). We see that the result for voting systems is very similar to that for recovery blocks: While for input conditions with low variances of version failure rates, both Model (2) and (3) yield acceptable predictions, Model (2) is far too pessimistic again for input conditions with largely varying version failure rates.

For the sake of brevity, the plots with the results of evaluation (c) are presented only for the two input conditions $S_{2,0}$ (lowest variance of version failure rates) and $S_{2,1}$ (highest variance of version failure rates): see Fig. 2 and 3. In both plots, the relative frequencies for "zero versions failing" have been omitted (they are just the differences of the sums of the other relative frequencies to unity) in order to get a better scaling. We have also omitted the results of Model (1) which again gave very unrealistic predictions. It can be seen that both for low and high variances of the version failure rates, the predictions of our model fit the true failure intensity distribution very well, whereas Model (2) yields a distribution decreasing too slowly. As to be expected, this effect is especially strong in the case of $S_{2,1}$.

As a final comment, let us mention that the observed "pessimism" of Model (2), where versions are assumed to have identical unreliabilities, gets understandable by looking at small examples: For simplicity, let us imagine three independently failing versions with failure rates $\pi_1 = 0.01$, $\pi_2 = 0.10$ and $\pi_3 = 0.19$, such that the average failure rate is $\bar{\pi} = 0.10$. The probability of a coincident failure of all these three versions is then 0.00019. If, however, $\pi_1 = \pi_2 = \pi_3 = 0.10$ is assumed, the probability of a coincident failure increases to 0.001. Replacing, in the unreliability calculation, diverse failure rates by their average has here the effect of a systematic overestimation of the unreliability of the system. This is very similar to a systematic trend to an over-pessimistic judgment of partition testing strategies, if calculations are based on *average* failure rates (see [9]).

	$S_{0,0}$	$S_{0,1}$	$S_{1,0}$	$S_{1,1}$	$S_{2,0}$	$S_{2,1}$
number of unrel. versions	6	10	7	8	8	8
Model (1) (indep.)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Model (2) (average π)	0.000012	0.000017	0.000004	0.000462	0.000006	0.002414
Model (3) (our model)	0.000001	0.000001	0.000003	0.000024	0.000003	0.000017
true value	0.000000	0.000000	0.000008	0.000010	0.000010	0.000028

Table III. Unreliability (= relative failure frequency) predictions and true unreliability values, both rounded to six decimal digits, for the recovery blocks built from the unreliable versions w.r.t. the six input conditions $S_{i,j}$.

	$S_{0,0}$	$S_{0,1}$	$S_{1,0}$	$S_{1,1}$	$S_{2,0}$	$S_{2,1}$
number of unrel. versions	6	10	7	8	8	8
Model (1) (indep.)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Model (2) (average π)	0.000185	0.000568	0.000029	0.010755	0.000137	0.052776
Model (3) (our model)	0.000078	0.000092	0.000051	0.000645	0.000068	0.000544
true value	0.000171	0.000343	0.000108	0.000447	0.000069	0.000300

Table IV. Unreliability (= relative failure frequency) predictions and true unreliability values, both rounded to six decimal digits, for the majority voting systems built from the unreliable versions w.r.t. the six input conditions $S_{i,j}$.

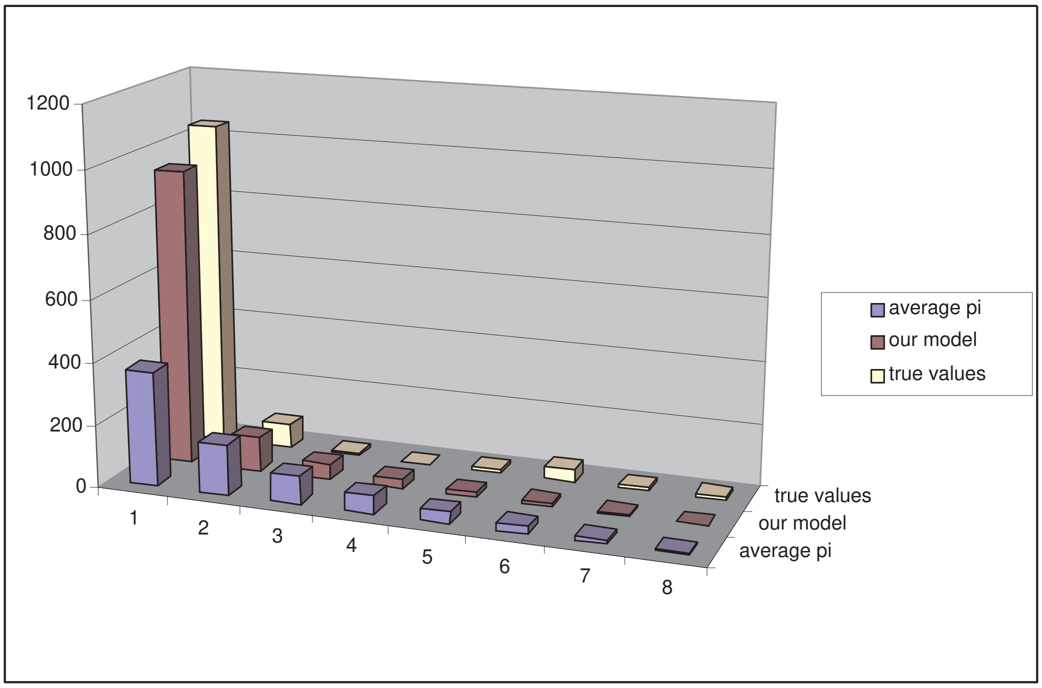

Fig. 2. Predictions and true values, in 10^{-6} units, for the relative frequencies of k versions failing together ($k = 1, 2, \ldots$), in case of input condition $S_{2,0}$. (Value for $k = 0$ omitted.)

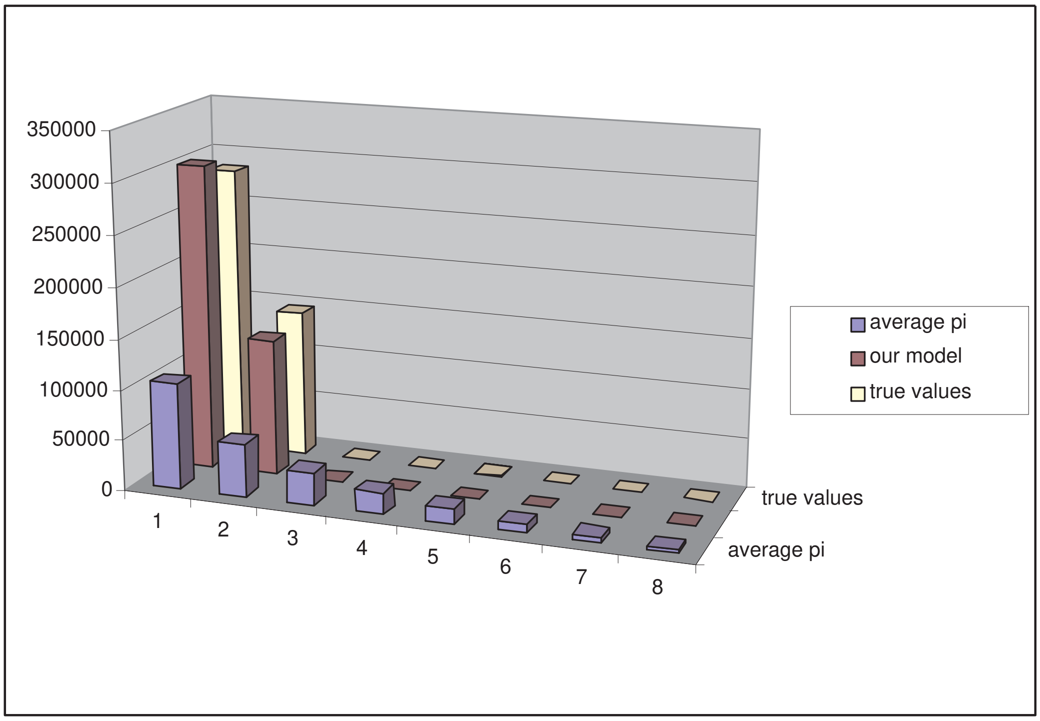

Fig. 3. Predictions and true values, in 10^{-6} units, for the relative frequencies of k versions failing together ($k = 1, 2, \ldots$), in case of input condition $S_{2,1}$. (Value for $k = 0$ omitted.)

5 Conclusion

We have developed a model for the reliability estimation of recovery blocks with fault-free test segments and of k-of-n voting systems, containing redundant versions with correlated failures. The approach overcomes a fundamental limitation of the well-known Eckhardt-Lee model by admitting that the version reliabilities can *differ*, while it still follows the suggestion by Nicola and Goyal to model the failure intensity distribution by a Beta-binomial distribution. We think that our approach might possibly also be extended to other modeling assumptions within the Eckhardt-Lee framework, e.g., estimating the intensity distribution from pairwise failure correlation, as it is suggested in [16]. This, however, is future work.

Our experimental results based on empirical data from [6] are promising, but further empirical investigations with a refined analysis of the outcome would of course be worthwhile.

An obvious next step in the development of the model might be the treatment of recovery blocks where the test segment cannot be assumed as fault-free anymore. For that purpose, the basic modeling strategy in [15] might be a guideline. Also, the input clustering effect, as studied in [4], should be taken account of.

A crucial problem in the application of our approach (and related approaches) is the estimation of version unreliabilities and of the correlation level θ. In this respect, both empirical and methodological work still has to be done. It would be very useful to have empirical results leading to a rule of thumb for getting reasonable estimates of θ depending on area of application and engineering techniques. Although reliability computations based on the Beta-binomial model seem to be relatively robust with respect to the exact value of

θ, its order of magnitude is of course important. As to the version unreliabilities, *software testing techniques* that provide quantitative reliability estimates would be extremely helpful. The author admits that most suggested techniques for this purpose (including his own in [7]) fail to be applicable for estimating the values π_i if the final test reveals no failures, but this drawback can possibly be overcome by Bayesian methods (see Miller et al. [12]).

Given that the indicated progress can be achieved, also the problem of *optimizing* the reliability of fault-tolerant systems under budget constraints, as addressed in [2], [3], [8] and other papers, can be attacked in the future at a new level of realism.

Acknowledgment. The author wants to thank P. Jedrzejowicz for helpful comments to a first version of this paper.

References

[1] Arlat, J., Kanoun, K., Laprie, J.-C., "Dependability modelling and evaluation of software fault-tolerant systems", in: *Predictably Dependable Computing Systems*, B. Randell, J. C. Laprie, H. Kopetz and B. Littlewood Ed., 441 – 457, Springer (1995).

[2] Belli, F., Jedrzejowicz, P., "An approach to the reliability optimization of software with redundancy", *IEEE Trans. Software Eng.*, vol. SE-17, 310 – 312 (1991).

[3] Berman, O., Ashrafi, N., "Optimization models for reliability of modular software systems", *IEEE Trans. Software Eng.*, vol. SE-19, 119 – 1123 (1993).

[4] Bondavalli, A., Chiaradonna, S., Di Giandomenico, F., Strigini, L., "Dependability analysis of iterative fault-tolerant software considering correlation", in: *Predictably Dependable Computing Systems*, B. Randell, J. C. Laprie, H. Kopetz and B. Littlewood Ed., 459 – 472, Springer (1995).

[5] Eckhardt, D. E., Lee, L. D., "A theoretical basis for the analysis of multiversion software subject to coincident errors", *IEEE Trans. Software Eng.*, vol. SE-11, 1511 – 1517 (1985).

[6] Eckhardt, D. E., Caglayan, A. K., Knight, J. C., Lee, L. D., McAllister, D. F., Vouk, M. A., Kelly, J. P. J., "An experimental evaluation of software redundancy as a strategy for improving reliability", *IEEE Trans. Software Eng.*, vol. SE-17, 692 – 702 (1991).

[7] Gutjahr, W. J., "Optimal test distributions for software failure cost estimation", *IEEE Trans. Software Eng.*, vol. SE-21, 219 – 228 (1995).

[8] Gutjahr, W. J., "Reliability optimization of redundant software with correlated failures", *Proc. ISSRE '98* (International Symposium on Software Reliability Engineering, Paderborn, Nov. 1998), 293 – 302 (1998).

[9] Gutjahr, W. J., "Partition testing vs. random testing: the influence of uncertainty", *IEEE Trans. Software Eng.*, vol. SE-25, 661 – 674 (1999).

[10] Kopetz, H., "The time-triggered approach to real-time system design", in: *Predictably Dependable Computing Systems*, B. Randell, J. C. Laprie, H. Kopetz and B. Littlewood Ed., 53 – 66, Springer (1995).

[11] Knight, J. C., Leveson, N. G., "An experimental evaluation of the assumption of independence in multiversion programming", *IEEE Trans. Software Eng.*, vol. SE-12, 96 – 109 (1986).

[12] Miller, K. W., Morell, L. J., Noonan, R. E., Park, S. K., Nicol, D. M., Murrill, B. W., Voas, J. M., "Estimating the probability of failures when testing reveals no failures", *IEEE Trans. Software Eng.*, vol. SE-18, 33 – 43 (1992).

[13] Nicola, V. F., Goyal, A., "Modeling of correlated failures and community error recovery in multiversion software", *IEEE Trans. Software Eng.*, vol. SE-16, 350 – 359 (1990).

[14] Saglietti, F., Ehrenberger, W., "Considerations on software diversity on the basis of experimental and theoretical work", in: *Software Fault Tolerance – Achievement and Assessment Strategies*, M. Kersken and F. Saglietti Ed., 21 – 32, Springer (1992).

[15] Saglietti, F., "A theoretical evaluation of the acceptance test in recovery block programming", in: *Software Fault Tolerance – Achievement and Assessment Strategies*, M. Kersken and F. Saglietti Ed., 213 – 223, Springer (1992).

[16] Tomek, L. A., Muppala, J. K., Trivedi, K. S., "Modeling correlation in sofware recovery blocks", *IEEE Trans. Software Eng.*, vol. 19, no. 11, 1071 – 1086 (1993).

APPENDIX

Proof of Proposition 3.2. As above, let Q denote the operational distribution on D. We have (as to the integral notation, see the remark after (4)):

$$b_k(k) = Pr\{\text{ all versions fail on } X\} = \int_D Pr\{\text{ all versions fail on } x\}\, dQ(x)$$

$$= \int_D \prod_{i=1}^{k} Pr\{\text{ version } i \text{ fails on } x\}\, dQ(x)$$

$$= \int_0^1 \prod_{i=1}^{k} Pr\{\text{ version } i \text{ fails on } x \mid p_i(x) = \varphi_i(p)\} \cdot f_0(p)\, dp.$$

The third equation holds because the failures are conditionally independent given $X = x$, and the last equation holds because

$$p_i(x) = \varphi_i(p) \quad \text{exactly if} \quad p_0(x) = p,$$

and the probability under Q of $p_0(X)$ being in $[p, p + dp]$ is just $f_0(p)dp$. By definition of $p_i(x)$,

$$Pr\{\text{ version } i \text{ fails on } x \mid p_i(x) = \varphi_i(p)\} = p_i(x) = \varphi_i(p),$$

which shows eq. (7).

It remains to demonstrate that the r.h.s. of (7) is invariant with respect to the basic distribution F_0. Let us replace F_0 by $\tilde{F}_0$, and let

$$\tilde{\varphi}_i = F_i^{-1} \circ \tilde{F}_0$$

be the counterpart to the function φ_i, based on $\tilde{F}_0$ instead of F_0 $(i = 1, \ldots, k)$. We want to show that

$$\int_0^1 \left(\prod_{i=1}^{k} \varphi_i(p) \right) f_0(p)\, dp = \int_0^1 \left(\prod_{i=1}^{k} \tilde{\varphi}_i(q) \right) \tilde{f}_0(q)\, dq, \tag{16}$$

where $\tilde{f}_0 = \tilde{F}_0'$. Let $\varphi_0 = \tilde{F}_0^{-1} \circ F_0$. Applying the transformation $q = \varphi_0(p)$ to the integral on the r.h.s., we obtain

$$\int_0^1 \left(\prod_{i=1}^{k} \tilde{\varphi}_i(q) \right) \tilde{f}_0(q)\, dq \;=\; \int_0^1 \left(\prod_{i=1}^{k} \tilde{\varphi}_i(\varphi_0(p)) \right) \tilde{f}_0(\varphi_0(p))\, \varphi_0'(p)\, dp. \tag{17}$$

Therein,

$$\tilde{\varphi}_i(\varphi_0(p)) = \left(F_i^{-1} \circ \tilde{F}_0 \circ \tilde{F}_0^{-1} \circ F_0\right)(p) = \varphi_i(p).$$

Furthermore, from $\tilde{F}_0 \circ \varphi_0 = F_0$, we get

$$\tilde{F}_0'(\varphi_0(p))\, \varphi_0'(p) = F_0'(p)$$

or

$$\tilde{f}_0(\varphi_0(p))\, \varphi_0'(p) = f_0(p).$$

By insertion into (17), eq. (16) is obtained. □