ISSN (e): **2250 – 3005** || Volume, 07 || Issue, 06|| June – 2017 ||
International Journal of Computational Engineering Research (IJCER)

Error Modeling in Dependable Component-based Systems

[1]debasish Pradhan, [2]ruchira Tripathy
Gandhi Institute of Excellent Technocrats, Bhubaneswar, India
Samanta Chandra Sekhar Institute of Technology and Management, Koraput, Odisha, India

ABSTRACT

Component-BasedDevelopment(CBD)ofsoftware,withitssuccessesinenterprisecomputing,hasthepromiseofbe-ing a good development model due to its cost effectivenessand potential for achieving high quality of components byvirtue of reuse.However, for systems with dependabilityconcerns, such as real-time systems, a major challenge inusing CBD consists of predicting dependability attributes,orprovidingdependabilityassertions,basedontheindivid-ualcomponentpropertiesandarchitecturalaspects.Inthispaper, we propose a framework which aims to address thischallenge. Specifically, we present a revised error classifi-cationtogetherwitherrorpropagationaspects,andbrieflysketch how to compose error models within the context ofComponent-Based Systems (CBS). The ultimate goal is toperformtheanalysisonagivenCBS,inordertofindbottle-necks in achieving dependability requirements and to pro-vide guidelines to the designer on the usage of appropriateerrordetectionandfaulttolerancemechanisms.

INTRODUCTION

ThemainadvantagesofCBDapproacharetheabilitytomanagecomplexityandthepossibilitytoselectthemostsuitablecomponentamongtheonesthatprovidesamefunc-tionality.However, the latter can be best achieved only ifthe design step incorporates rigorous analysis for this spe-cific need. This issue becomes all the more relevant whenCBD is used for developing dependable systems, since onehastoanalyzemultipleextra-functionalpropertiesaswell.

Our main goal is development of a framework based onwell-founded theories, while keeping industrial realities infocus, which will provide meaningful reasoning about de-pendability attributes in CBS based on the characteristicsof the component model, properties of individual compo-nentsandcomponentconnectionschemeinagivendesign.Sinceerrorsareoneofthemainimpedimentsforachieving dependability, this paper particularly focuses on modelingtheerrorbehaviorofcomponentsanderrorpropagationas-pects in order to reason about the dependability attributesof the composed system and its failure modes. We use anin-housedevelopedcomponentmodel(SaveCCM)[2]toil-lustrate how a specific component model can influence theerrorpropagationaspects.

In a recent work, Elmqvist and Nadjm-Tehrani [7] ad-dressed formal modeling of safety interfaces and providedcompositional reasoning about safety properties of com-posed systems. Our focus is more on reliability and timingaspects and on analytical approaches.Grunske and Neu-mann [9] have proposed an approach to model error be-havior of composed systems by using the Failure Propaga-tionandTransformationNotation(FPTN)foreacharchitec-turalelementandtoconstructthecomposedsystems'Com-ponent Fault Trees (CFT) from the FPTN models to per-form safety analysis. Rugina et al. [14] proposed a frame-work where the Architecture Analysis and Design Lan-guage (AADL) with the features of Error Model Annex isusedtocreatemodelsofcomposedsystems'errorbehavior.Then, these models are converted to GeneralisedStochas-ticPetriNets(GSPNs)orMarkovChainstobeanalyzedbyexisting tools.More recently, Joshi et al.[11] have pro-posedanapproachtoconverterrormodels,generatedusingAADL with Error Model Annex, to Fault Trees to performfurtheranalysis.

Asubstantialamountofresearchhasbeenconductedonreliability modeling of composed systems based on indi-vidual component reliabilities, with a recent focus on ar-chitecture based models.Most of these works assume theexistence of known probabilities for error state transitions,and only a few address the error propagation aspects.Onthe other hand, research on dependable systems has beenfocussing more on fundamental system level models of er-rors,andmechanismsfortoleratingthoseerrormodes,witharguablylessinterestonhowthesemodelsarelinkedtothereliabilitypredictionmodels.Inourview,thelinksbetweenthese two research directions are loosely coupled and lessexplored. Specifically in CBD, architectural decisions andspecificaspectsofthecomponentmodelwillinfluencethe dependability evaluations. Our aim is to enable end-to-endlinkingfromsystemleveldependabilityrequirements(nor-mally specified in terms of diverse qualitative/quantitaiveterms), to models for dependability evaluation and predic-tions of composed systems.We envision our research toprovide substantial clarity and simplifications needed forCBDofapplicationswithdependabilityconcerns.

The rest of the paper is organized as follows: in Section2 we state the challenges in system level modeling of errorbehavior, and present the principal parts of our proposedframework.Section 3 presents our revised error classifi-cation from a CBS perspective.Section 4 discusses errorpropagationandcompositionaspects,whicharefurtherex-emplifiedinSaveCCM,andSection5presentsconclusionsandongoingresearch.

1. Outlineoftheproposedframework

The major challenges in realization of a generalizedframeworkfordependabilityevaluationofCBSare:

- diversityofdependabilityrequirementsspecification

differentdependabilityattributesrequiredifferentanalysistechniquesandapproaches

- limitedinformationoncomponentproperties

lackoftechniquesforperforminganalysiswithpartialorevolvinginformation

relating usage profiles of components to target systemcontexts

non-scalabilityofmostoftheformalanalysistech-niquestoindustrial-sizesystems

In order to enable modeling and analysis of system-level dependability behavior, the framework must includedependability requirements specification, component-levelerror modeling, and system-level dependability analysis,whicharebrieflymentionedinfollowingsubsections.

Dependabilityrequirementspecifica-tions

At this step, the system designer has to specify the de-pendability requirements for the target system. Due to thediversity of the dependability attributes, as well as the var-ied industrial priorities and practices, this step is critical asithasaconsiderableimpactonthesubsequentanalysis(in-cluding the choice of techniques).For instance, the reli-ability requirements of systems are usually defined in di-verseterms,rangingfromqualitativetoquantitativeones.

Atypicalrequirementspecificationcanbe'Systemreliabil-ityshouldexceed0.99999'or'Systemshouldnothaveanytiming failures even under a hardware node failure'.Theframework must have means to accurately capture and for-mally specify a wide variety of such requirements, whichthesubsequentanalysistechniquesneedtoaddress.

While designing a dependable system, the goal is typi-cally to achieve fail-controllability [3], i.e., to introduce acertain degree of restrictions on how the system can fail.The level and type of such restrictions are usually dependentontheapplicationdomain,criticalityofthesystem,andthedependability attributes that are considered.Typical fail-ure modes include fail-operational, fail-safe, fail-soft, fail-silent,fail-stop,crashandByzantine(arbitrary)failures[3].Failuremoderequirementscaneffectivelybeusedforgen-erating subsystem-level requirements in a hierarchical wayandcanhelpinperforminglocalizedanalysis.

Component-levelerrormodeling

Typically, this step involves modeling error behavior ofindividual software components, as well as other systemelements, such as component connectors, hardware nodes,middleware, and communication media. Our plan is to useprobabilistic automata with timing, where nodes of the au-tomata represent error states, and edges denote transitionprobabilities.An approach based on AADL [14] can besuitable for this step, with proper extensions on the errormodelingaspects.Ourintegrateddevelopmentenvironmentfor CBS is being designed to specify and include informa-tion about component error behaviors with varying levelsof details, based on the available specifications. The levelof details in component-level error models, as well as thedependabilityrequirementspecificationsofthesystem,willdecidethechoiceofanalysistechniquetobeperformed.

System-leveldependabilityanalysis

The analysis to be performed at this step depends on thedependability specifications and the component-level errormodels.Our aim is to get the basic structure in place sothatmultipleanalysistechniquescanbeeasilyintegratedtoour framework. A challenging issue is how to compose er-ror models to obtain a system-level error behavior.Errorpropagation can occur between two components,betweena component and another system element, or between twosystem elements. The architecture of the system will serveas an input to this step, where both impact and criticalityanalysis will be performed.Ideally, by looking at the er-ror model of the composed system, one should be able toobserve whether the system can possibly fail in a certainmode that is not allowed. If this is the case, the frameworkshouldfurtherenabletheidentificationofthecriticalpathsin the architecture, and provide guidelines for efficient de-tection/recovery/correction strategies along with appropri-atelocationforincorporatingthem,sothattheresultingsys-temmeetstheoriginaldependabilityrequirementsasspeci-fiedbythesystemdesigner.

Figure1illustratestheskeletonofourproposedmethod-ologyforcomposingerrorinformationtoperformasystem-level error analysis.The methodology consists of criticalpath identification followed by propagation analysis per-formed on each identified path where the type of analysisdepends on the specific failure mode requirement. Thoughcomponents are usually considered as black boxes, we as-sumetraceabilityofacriticalparameterevaluationthroughthe component chain. If, on the other hand, this is not pos-sible,wemayhavetoconsiderallpossiblescenarios.

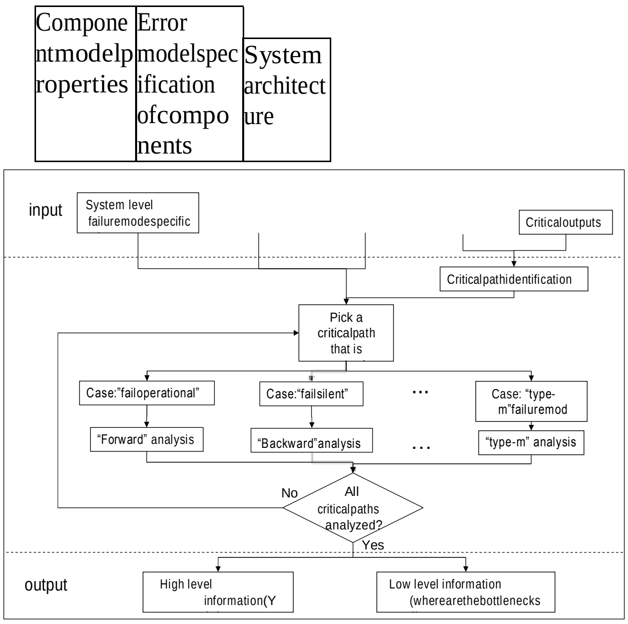

Figure1.System-levelerroranalysis

Domain

Incomponentbasedsystems,outputsgeneratedbycom-ponents can be specified by two domain parameters, viz.,value and time [3, 4, 13]. Our hypothesis is that toleratingvalueandtimingerrorsatcomponent-level,requiresdiffer-entapproacheswithsignificantlydifferentassociatedcosts.Hence, separation of value and time domains will enablethe use of dedicated fault tolerance mechanisms for eachtype, as well as aid in achieving better error coverage withminimumcost.Inthispaper,wedefinethespecifiedoutputgeneratedbyacomponentasatuplebasedonthesedomain parameters:

SpecifiedOutput(SO)= $<v^*,V,T,\Delta_1,\Delta_2>$

where the v*is the exact desired value, Vis the set of ac-ceptable values, Tis the exact desired point in time whenthe output should be delivered and $[T\Delta_1, T + \Delta_2]$ is theacceptabletimerangefortheoutputdelivery.

Theoutputgeneratedbyacomponentisdenotedas:GeneratedOutput(GO)=<v,t> wherevisthevalueandthetisthetimepointwhenthe outputisactuallydelivered.

TheGOisconsideredtobecorrectif:

$v \in V$and$T-\Delta_1 \leq t \leq T+\Delta_2$

Value errors: The output generated by a component iserroneous in value domain (e_v) if v /V , where Vis theset of acceptable values.We first classify errors in valuedomainas**subtle**(e^s),and**coarse**(e^c)basedonourknowl-

v v

2. Errorclassification-revised

The error characteristics presented in this section arebasedonasynthesizedviewofseveralworks[3,13,4,10, 6].We follow the basic classification of Avizieniset.al.[3] while extending it into details with other works,mostofwhichaddressnarrowerareasbutwithfinerdetails.It also presents various aspects of errors in two categoriesbasedontheirinfluenceontheerrorhandlingmechanisms.Thesecategoriesessentiallydetermine'whichmec hanisms'and 'how much' are needed for adequate error handling.Thevariousaspectsconsideredaredomain,consistency,de-tectability,impact,criticality,andpersistenceoferrors.Thedomain and consistency determine what kind of error han-dling mechanisms are appropriate while the rest determinetheamountoferrorhandlingneeded. edgeaboutthesetofreasonablevaluesfortheoutputandthesyntaxthatshouldbefollowedasin[4,13].

- Inexactvalueerrors(e^e)

$v \notin \bar{V}$,where$\bar{V}=\{v^*\}$

- Unacceptabledistinctvalueerrors(e^d)

$v \notin \bar{V}$, where $\bar{V}=\{v^*, v_1, v_2, ..., v_n\}$, $v^*$is the idealvalueand$v_1,v_2,...,v_n \in$ theotheracceptablevalues

- Inaccuratevalueerrors(e^a)

$v \notin \bar{V}$, where $\bar{V} = \{v^*\Delta^v, ..., v^*1, v^*, v^*+1,...,v^*+\Delta^v\}and[v^*\Delta^v_1, v^*+\Delta^v_2]$istherangeofacceptablevalues

Avalueerrorisacombinationoftheaboveclassifica-tions,i.e.,e^{xy}_v,where$x\in\{c,s\}$and$y\in\{a,d,e\}$.

Timing errors: In [4, 13, 6] and in our classification, er-rors in time domain are classified into early, late and in-finitelylate(omission)timingerrors.

- earlytimingerrors(e^e_t):$t<T-\Delta_1$
- latetimingerrors(e^l_t):$t>T+\Delta_2$
- omissiontimingerrors(e^o_t):$t=\infty$

Additional classes [13] are, bounded, omission, and per-manent omission (crash or permanent halt) errors.paragraph Errors in both time and value domain:Componentoutputsunderthiscategoryareerroneousinbothvalue andtimedomainsimultaneously,i.e.,$e^{a,b}_{v,t}$,wherea $\in$ $\{ce,cd,ca,se,sd,sa\}$andb$\in\{e,l,o\}$if:

$v \notin \bar{V}$and$(t<T-\Delta_1$or$t>T+\Delta_2)$

Consistency

If a component provides replicas of an output to sev-eral components,consistency issues may arise.In thiscase,the errors are considered consistent if all receiversget identical errors.In [13], multi-user service errors areclassified into consistent value errors, consistent timing er-rors,consistentvalueandtimingerrors,andsemi-consistentvalue errors. In semi-consistent value errors, some out-put replicas have unreasonable,or out-of-syntax values,whiletheresthaveidenticallyincorrectvalues.In[4],non-homogeneousoutputreplicasaredefinedtobeerroneous.

- Inconsistent errors: Replicas of an output are definedas inconsistent if there are both correct and incorrectreplicas.

The characteristics presented so far define our error classi-fication and will be used in both propagation analysis andcomposition of component error models.Furthermore theclassification will be used to determine which error han-dlingmechanismsareadequatetocontroltheerrorbehaviorduringcomposition.

3. ErrorpropagationinCBS

Errors in a component based system can occur in soft-ware components, middleware, or hardware platform, andcan propagate up to a system interface causing a systemfailure with a certain probability. This probability is, in itsturn, dependent on the probability of error occurrences, theisolation between different system elements, existing errordetection and handling mechanisms, as well as the type oferrors. The research effort is currently increasing for find-ing ways to get these probabilities, and to use them appro-priately[10,8,12,1,5].

We define the set of errors E, which includes instantia-tionsoferrortypesdiscussedinthepreviоussection.WealsodefinethefollowingsubsetsofEasfollows:

$E^{in}$is the set of errors that are propagated into com-ponentC_i

• i

Anon-homogeneousvalue(ortiming)erroroccursifthe

• E

geni

isthesetoferrorsthatareinternallygener-

values(ortimes)ofreceivedoutputsarenotcloseenough toeachother.Closenessisspecifiedbyusingthresholdval-ues. In our classification, we use both consistency and ho-mogeneityconcepts.

• Consistenterrors:Replicasoftheoutputfromacom-

ponentareconsistentlyerroneousiftheybelongtothe

atedbycomponent$C_i$andpropagatedout withoutany changes

E_i is the set of errors propagated into $C_i$thatare propagatedoutwithoutanychanges

• pass

• $E^{mod}$is theis thesubsetofE^{in}thataretransformed

i i

sameerrorcategory,e.g.,bothhavecoarsevalueerrorsorlatetimingerrors.

Wefurtherclassifytheseerrorsas:

– Precise: The values or generation times of repli-casareconsistentlyerroneousaswellasbotharewithinaprecisionrangeoridentical.

– Imprecise:Thevaluesortimingofreplicasare

toanothererrortype,maskedorcorrected

E^{trans}isthesetoferrorsthatwereoriginallybelong-ingtoE^{mod}orinternallygeneratederrorsandtrans-formedintothemembersformingthisset

• i

$E^{out}$is the set of errors that are propagated from com-ponentC_i i

consistentlyerroneous.Howevereithervaluesor

• i

generationtimes(dependingontheerrortype) i i i

E^{in}

$= E^{mod} \cup E^{pass}$

E^{out}_i areoutsidethespecifiedprecisionrange.

$= E^{gen}_i \cup E^{pass}_i \cup E^{trans}_i$

• Semi-consistent errors: Replicas of an output are de-fined as semi-consistently erroneous if all users receiveerroneousoutputswhileatleastoneofthembelongstoadifferenterrorcategorythantheothers.

ErrorscanbetransformedintoE^{trans}byeitherC_i'snor-mal execution or by error handling mechanisms.Thesemechanisms can be implemented within components atcomponentdesignstage, atthe componentinterfacesat the

architectural design, or at integration stages of CBD. Var-ious mechanisms for different types of errors and their ef-fects on error propagation are discussed in the followingparagraphs.

Transformation of value errors: The possible ways oferrortransformationsinvaluedomainareshowninTable1.

Cause	Initialerror	Finalerror
Errordetection	e_v	e_v(transformationi ntimedomain)
Errormasking	e_v	noerror
Errorcorrection	e_v	noerror
Componentoperation	e_v e^s	noerror e^c

	e_v^c	e_v^s

Table1.Transformationsofvalueerrors

One way to detect coarse value errors is using reason-ableness checks. Implementing reasonableness checks ne-cessitates having knowledge about the behavior of the pro-ducer,forexample,arangecheckingmechanismmarksthetemperaturereadingofaroomaserroneousifthevaluereadis200°C based on our knowledge about the reasonableboundaries for that output.Coding checks are used to de-tectnon-codevalueerrorswhichisaspecifictypeofcoarsevalue errors (parity-check is an example for this type ofcheck). Obviously, if more advanced error detection mecha-nismsareused,whichcanidentifymorecomplexerroneousbehaviors, the coverage of detectable errors is increased.Detecting subtle value errors is performed by more expen-siveerrordetectionmechanisms,suchasreplicacheckingatavoterelement.Propagationofvalueerrorscanbeblockedafterdetection,bysimplynotallowingtheerroneousoutputto be delivered to the next component. In this case a valueerroristransformedintoanomissiontimingerror.
Certain means allow masking of value errors, such asN-modular redundancy techniques, while some others cancorrect value errors by using, e.g., error correction codes.Both masking and correction techniques enable continua-tionofcorrectfunctioninguponerrors.

Transformationoftimingerrors: Errorsintimedomaincanbetransformedaccordingtothefollowingorder:

$e_t^e \rightarrow$ noerror $\rightarrow e_t^l \rightarrow e_t^o$

Timing checks and watchdog timers can be used to de-tect timing errors produced by components.Early timingerrors can be corrected by introducing delays. Propagationof early or late timing errors can be blocked by not trans-mittingthem,iftherearenomeanstocorrectthem.Insuch cases, these errors are transformed into omission er-rors.When a value error is detected and omitted, as de-scribed previously, the output is actually transformed fromhavingnotimingerrortoanomissionerror.
For errors regarding consistency, similar checks can beusedandinconsistenterrorscanbetransformedintoconsis-tenterrorsinbothvalueandtimingdomains.
To illustrate how a specific component model can in-fluencetheerrorpropagationaspects,wehaveconsid-eredthein-housedevelopedSaveCompComponentModel(SaveCCM)[2]andpropagationbetweencomponentsthroughconnectors.

SaveCCMcomponentmodel

SaveCCM was developed under the SAVE project andwasintendedforuseinautomotiveapplications.InSaveCCM, systems are built by composing entities whichbelongtooneofthreemaincategories,namelycomponents,switchesandassemblies,viawell-definedinterfaces.
Components are basic entities in SaveCCM that followstrict read-execute-write semantics.A component is ini-tially in an inactive state.Once all input trigger ports areactivated,inputdataportsarereadandthecomponentstartsexecuting.When the execution is completed, results arewritten to output data ports, input data ports are reset, andall output trigger ports are activated. Then the componentreturnstotheidleorinactivestate.Switchesarelightweightcomponents that allow changing the interconnections ofcomponents either statically, for offline configuration, ordynamically at run-time.Switches are not triggered andonly perform routing of incoming data to output ports ac-cording to the connection pattern guards.Finally, assem-bliesareencapsulatedsubsystemswhoseinternalstructuresmay(ormaynot)bevisiblefromtherestofthesystem.
Interfaces between SaveCCM entities consist of inputandoutputports.Theyarefurtherclassifiedintodataports,trigger ports, and both data and trigger ports. Connectionsbetween components consist of immediate or complex con-nections, where immediate connections are assumed to be-have as ideal connections which take place instantly withoutanylossofinformation.Complexconnections,ontheotherhand,areusedtomodelmorerealisticconnectionscenarios,e.g.,withcertaindelaysandpossiblelossofinformation.

ErrorpropagationinSaveCCM

Inthissection,wefirstinvestigatewhicherrortypescanbe propagated from one entity to another through differentSaveCCM ports (Table 2).If two SaveCCM entities areconnected by a trigger port, then the preceding entity canpropagate only timing errors by triggering (therefore acti-vating)thefollowingentityatanincorrecttime.Ifthecon-

	ValueErrors	TimingErrors
Dataports	C	C
Triggerports	-	C
Dataandtriggerports	C	C

Table2.ErrorpropagationthroughSaveCCM

nection is implemented with a data and trigger port, bothvalue and timing errors can be propagated. This is also thecase for connections implemented with a data port, sincethe time when the data is written to the port will determineif there is a timing error. Hence, SaveCCM entities can beclassified with respect to error generation and propagationasfollows:

- SaveCCMentitiesthatcangeneratevalueerrors:These are the entities that have output data or outputdataandtriggerports.
- SaveCCMentitiesthatcanpropagatevalueerrors:En-tities in this group have input data or input data andtrigger ports to receive a value error from a preced-ing entity. Furthermore they must have output data oroutputdataandtrigger portsinordertopropagatetheerrortothefollowingentity.
- SaveCCM entities that can generate timing errors: Anyentitythathasoutputportscangeneratetimingerrors.
- SaveCCM entities that can propagate timing errors:Similarly any SaveCCM entity can propagate timingerrors provided that there exists at least one input andoneoutputport.

As different component models have different levels ofimpact on the error propagations, similar detailed analysis,inthecontextofthegivencomponentmodel,isessentialforanaccurateandcomputationallyfeasiblesystemlevel errorbehaviorprediction.

4. SummaryandOngoingWork

In this paper, we have proposed a framework to enablecompositional reasoning of error models. We have surveyedvarious error classifications and failure modes in the litera-ture with the aim of identifying their relations/contrasts aswell as in arriving at an ’all-encompassing compilation ofclassifications’.WehaveinvestigatedtheerrorpropagationinCBSanddiscussedtheeffectsoferrorhandlingmecha-probabilisticvariantsofthem),b)providelinkstoarchitec-tural reliability prediction models together with new theo-ries on dependability reasoning of multi-level compositions,as well as c) instantiate our framework on the SaveCCMsuccessor,i.e.,ProComp,currentlyunderdevelopment.

REFERENCES

[1] W.Abdelmoez,D.Nassar,M.Shereshevsky,N.Gradetsky,R. Gunnalan, H. Ammar, B. Yu, and A. Mili.Error prop-agation in software architectures.Proceedings of the 10thInternationalSymposiumonSoftwareMetrics,2004.

[2] M.Akerholm,J.Carlson,J.Fredriksson,H.Hansson,J.Hakansson,A.Möller,P.Pettersson,andM.Tivoli.The SAVEapproachtocomponent-baseddevelopmentofvehic-ular systems. Journal of Systems and Software, 80(5):655–667,2007.

[3] A.Avizienis,J.Laprie,andB.Randell.Fundamentalcon-cepts of dependability.Research Report N01145, LAAS-CNRS,April2001.

[4] A. BondavalliandL.Simoncini.Failureclassificationwithrespecttodetection.Proceedingsof2ndIEEEWorkshoponFutureTrendsinDistributedComputin,1990.

[5] V.CortellessaandV.Grassi. Amodelingapproachtoanalyze the impact of error propagation on reliability ofcomponent-based systems. Proceedings of the Component-basedSoftwareEngineeringConference,2007.

[6] K.EchtleandA.Masum.Afundamentalfailuremodelfor fault-tolerantprotocols.ProceedingsofComputerPerfor-manceandDependabilitySymposium,2000.

[7] J.ElmqvistandS.Nadjm-Tehrani. Safety-orientedde-signofcomponentassembliesusingsafetyinterfaces.Elsevier-ElectronicNotesinTheoreticalComputerScience,(182):57–72,2007.

[8] L.Grunske.Towardsanintegrationofstandardcomponent-based safety evaluation techniques with SaveCCM.Pro-ceedings of the Conference Quality of Software Architec-tures,2006.

[9] L.GrunskeandR.Neumann.Qualityimprovementbyin-tegrating non-functional properties in software architecturespecification.ProceedingsoftheSecondWorkshoponEval-uatingandArchitectingSystemdependabilitY,2002.

[10] M.Hiller,A.Jhumka,andN.Suri.EPIC:Profilingtheprop-agationandeffectofdataerrorsinsoftware.IEEETransac-tionsonComputers,53(5):512–530,2004.

[11] A.Joshi,S.Vestal,andP.Binns.Automaticgenerationofstaticfaulttreesfromaadlmodels.ProceedingsoftheWork-shoponArchitectingDependableSystems,2007.

[12] P. Popic, D. Desovski,W. Abdelmoez,and B. Cukic.Errorpropagation in the reliability analysis of component basedsystems.Proceedings of the 16th IEEE International Sym-posiumonSoftwareReliabilityEngineering,2005.

[13] D.Powell.Failuremodeassumptionsandassumptioncov-erage.Proceedingsof22ndInternationalSymposiumon