# Formal Modeling Self-Adaptive Intelligent Service Component Based on Extenics

https://doi.org/10.3991/ijoe.v12i12.6452

R. Fan[1], Y.F.Chen[1], S.C.Cao[2], G.B.Lei[1] and C. Yue[1]
[1] Guangdong Ocean University, Zhanjiang, China
[2] University of Illinois at Urbana-Champaign, Champaign, USA.

***Abstract*—After thirty years, Extenics made great progress in innovation theory, formal method and real application. Through the formalized research for its logical cell basic-element, the information-knowledge-strategy formalized system, extension analysis methods, extension transformation methods and superiority evaluation methods are set up. We introduce Extenics theory to describe, analyze and evaluate self-adaptive nature, scope and extent about the intelligent service component. First, software entities are defined by basic-element, the intelligent service component is structured into basic-element net, which is analyzed, transformed and evaluated by method of combining qualitative with quantitative. Final, intelligent service component formal model is established.**

***Index Terms*—Extenics theory, formal modeling, intelligent service component, self-adaptive software.**

## I. INTRODUCTION

Extenics is an original and traverse discipline put forward by Chinese scholars in 1983[1]. It discusses the possibility of matters' extension and rules and methods for innovation with formalized models, which are used to solve contradictory problems. After more than 30 years of development, Extenics made great progress in innovation theory, formal methods and real application [2, 3]. Using flexible thinking of "changing insensible to knowable", "changing infeasible to feasible" etc., set up Extenics innovation method system [4]. Extenics theory and Extenics innovation methods have used to specific application areas, formed extension engineering method, such as extension detecting method, extension control method, extension planning method, extension marketing method[3]. Around Extencis software development, proposed Extencis information-knowledge-strategy system[5]. Extension strategy generating system, extension data mining methods, extension design methods have been constructed [3].

The most active research on self-adaptive software entities are component, service and agent. Component assembles software system by providing external interfaces [6, 7]. Service builds larger, loose component by service composition [8, 9]. By perceiving, reasoning and optimizing, agent actively responds to external changes [10, 11]. Existing software formal methods can't described activities to solve the conflict between entities, entity and environment by continuous evolution, even can't quantitatively analyze and verify self-adaptive type, scope, extent and trends of dynamic evolution, it is an issue which self-adaptive formal method to solve.

We discuss feasibility of using Extenics theory to describe, analyze and evaluate self-adaptive nature, scope and extent about intelligent service component [12]. Its internal entities are defined by basic-elements, the intelligent service component is structured based on basic-element net, then self-adaptability is analyzed, transformed and evaluated by methods of combining qualitative with quantitative. Final, intelligent service component formal model is established.

## II. EXTENICS FORMAL MODEL OF INTELLIGENT SERVICE COMPONENT

We use the intelligent service component as example to illustrate the Extenics formal methods.

1 first of all, all levels of internal entities and their relations of intelligent service components are represented by figure 1.

2 Composite-elements represent semantic information of internal entities. It consists of identity ID, interfaces I, controls C, events E, goals G, plans P, knowledge base K, services S and so on internal entities. All internal entities basic-elements are shown in figure 2.

3 Using Base-elements to replace internal entities, an intelligent service component base-elements diagram (ISC BED) is shown in figure 3.

4 The relation between element Bi and Bj is represented by the relation-element R, and are one-way relation, is two-way relation, the figure 3 can be represented by the formal base-element net in figure 4. It gives rich relations semantic of the intelligence service component.

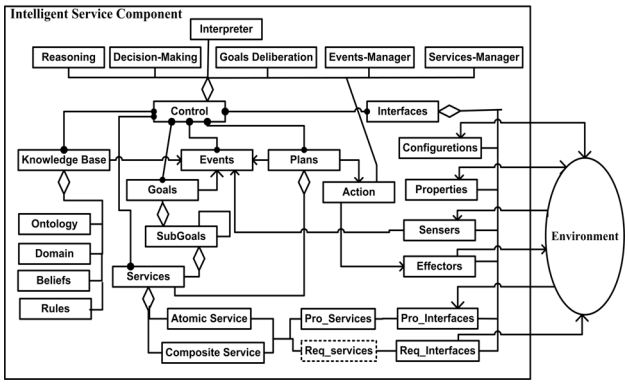


Figure 1. Intelligent Service Component Elements and Relations

$$\begin{bmatrix} ISC & Id & ID \\ & Interfaces & I \\ & Controls & C \\ & Events & E \\ & Goals & G \\ & Plans & P \\ & Knowledge\ Base & K \\ & Services & S \end{bmatrix}, I = \begin{bmatrix} Interfaces & Pro\_Interfaces & I1 \\ & Req\_Interfaces & I2 \\ & Sensors & I3 \\ & Effectors & I4 \\ & Properties & I5 \\ & Configurations & I6 \end{bmatrix}, C = \begin{bmatrix} Controls & Knowlege\text{-}manager & C1 \\ & Events\text{-}manager & C2 \\ & Interpreter & C3 \\ & Goals\text{-}Deliberation & C4 \\ & Services\text{-}manager & C5 \\ & Plans\text{-}manager & C6 \\ & Action & C7 \end{bmatrix}$$

$$E = \begin{bmatrix} Event & InnerEvent & E1 \\ & OutsiadEvent & E2 \end{bmatrix}, G = \begin{bmatrix} Goal & Id & ID \\ & SubGoal & SG \end{bmatrix}, P = \begin{bmatrix} Plan & Id & ID \\ & PreCondition & P1 \\ & Body & P2 \\ & PostCondition & P3 \end{bmatrix}$$

$$K = \begin{bmatrix} Knowledge\ Base & Rules & K1 \\ & Belifes & K2 \\ & Ontology & K3 \\ & Domain & K4 \end{bmatrix}, S = \begin{bmatrix} Services & Req\text{-}Services & S1 \\ & Pro\text{-}Services & S2 \\ & Com\text{-}Services & S3 \\ & Atom\text{-}Services & S4 \end{bmatrix}$$

Figure 2. Intelligent Service Component Basic-Elements

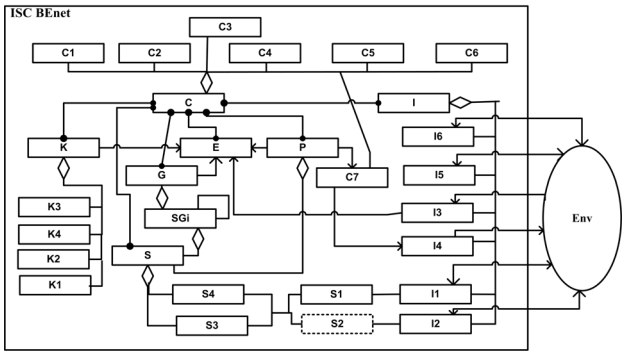


Figure 3. Intelligent Service Component Basic-Elements Diagram

5 Static semantics analysis:

1) Control relation ( $\overrightarrow{R}_{control}$ ):i.e., $\overrightarrow{R}_{control}(C3,(C1,C2,C4,C5,C6,C7))$.

2) Assemble relation ( $\overrightarrow{R}_{assemble}$ ): i.e., $\overrightarrow{R}_{assemble}(K,(K1,K2,K3,K4))$.

3) Realization relation ( $\overrightarrow{R}_{implement}$ ): i.e., $\overrightarrow{R}_{implement}(SGi,(Si,i=1,2,...,n))$.

4) Production relation ( $\overrightarrow{R}_{production}$ ), i.e. $\overrightarrow{R}_{production}(E,(I,K,G,S,P))$.

5) Input relation ( $\overrightarrow{R}_{input}$ ), i.e., $\overrightarrow{R}_{input}(I3,Env)$.

6) Output relation ( $\overrightarrow{R}_{output}$ ), i.e., $\overrightarrow{R}_{output}(I4,Env)$.

7) Provide service relation ( $\overrightarrow{R}_{pro-service}$ ), i.e., $\overrightarrow{R}_{pro-service}(I1,ISCs)$.

8) Request service relation ( $\overrightarrow{R}_{req-service}$ ), i.e., $\overrightarrow{R}_{req-service}(I2,ISCs)$.

9) Link relation ( $\overrightarrow{R}_{link}$ ), i.e., $\overrightarrow{R}_{link}(I1,S1)$, $\overrightarrow{R}_{link}(I2,S2)$.

10) Set / get relation ( $\overrightarrow{R}_{set/get}$ ), i.e., $\overrightarrow{R}_{set/get}(I5,Env)$.

11) Configuration relation ( $\overrightarrow{R}_{config}$ ), i.e., $\overrightarrow{R}_{config}(I6,Env)$.

$$C3(\overrightarrow{R}_{control}) \begin{cases} C1(\overrightarrow{R}_{control})K(\overrightarrow{R}_{assembel}) \begin{bmatrix} K1 \\ K2 \\ K3 \\ K4 \end{bmatrix} \\ C2(\overrightarrow{R}_{control})E(\overrightarrow{R}_{production}) \begin{bmatrix} I \\ K \\ G \\ P \\ S \end{bmatrix} \\ C4(\overrightarrow{R}_{control})G(\overrightarrow{R}_{assembel}) \left[ SQi(i=1,2,...,n)(\overrightarrow{R}_{implement})S \right. \\ C5(\overrightarrow{R}_{control})S(\overrightarrow{R}_{assembel}) \begin{bmatrix} S1(\overrightarrow{R}_{link})[I1 \\ S2(\overrightarrow{R}_{link})[I2 \\ S3(\overrightarrow{R}_{assembel})[S4 \\ S4 \end{bmatrix} \\ C6(\overrightarrow{R}_{control})P(\overrightarrow{R}_{assembel})S \\ C7(\overrightarrow{R}_{control}) \begin{cases} P(\overrightarrow{R}_{assembel})[S \\ I(\overrightarrow{R}_{assembel}) \begin{bmatrix} I1(\overrightarrow{R}_{Pro-service})[ISCs \\ I2(\overrightarrow{R}_{Req-service})[ISCs \\ I3(\overrightarrow{R}_{input}) \left[ Env(\overrightarrow{R}_{assembel}) \begin{bmatrix} ISCs \\ Envs \end{bmatrix} \right. \\ I4(\overrightarrow{R}_{output}) \left[ Env(\overrightarrow{R}_{assembel}) \begin{bmatrix} ISCs \\ Envs \end{bmatrix} \right. \\ I5(\overrightarrow{R}_{set/get}) \left[ Env(\overrightarrow{R}_{assembel}) \begin{bmatrix} ISCs \\ Envs \end{bmatrix} \right. \\ I6(\overrightarrow{R}_{config}) \left[ Env(\overrightarrow{R}_{assembel}) \begin{bmatrix} ISCs \\ Envs \end{bmatrix} \right. \end{bmatrix} \end{cases} \end{cases}$$

Figure 4. Intelligent Service Component Formal Basic-elements Net

6 Dynamic semantic analysis:

We introduce affair-element A to represent actions or functions, for example, the B1A1B2A2B3... Bi-1Ai-1Bn represents a series of element Bi use successively actions or functions Ai act to Bi+1. In addition, the operator •, |, + respectively express in subsequent, parallel and choice of semantics. The dynamic semantic descriptions are as follows:

1) Interpreter C3 controls knowledge base manager C1 initializes the knowledge base K, C1 reads information

related domain ontology K3, domain rules K1 and domain application K4 and write to the belief base K2, complete the initialization tasks. Extenics semantics is as below:

$$C3\vec{A}_{control}(C1\overleftarrow{A}_{read}(\mathrm{K1},\mathrm{K3},\mathrm{K4}).C1\overline{\mathrm{A}}_{write}K2)$$

2) Environmental Env change produces events E through interface I3 input to the event queue E. Extenics semantics is as below:

$$Env\vec{A}_{producte}E\vec{A}_{input}I3\vec{A}_{input}E$$

3) The interpreter C3 controls the event manager C2 to read event e from event queue E, and current cognitive of K2, then adopts rhombus thinking, translates to goal G for response to the event e. Extenics semantics is as below:

$$C3\vec{A}_{control}(C2\overleftarrow{A}_{read}(\mathrm{E},\mathrm{K2}).\mathrm{C2}\overline{\mathrm{A}}_{r\,\mathrm{hom}bus}(\mathrm{e},\mathrm{K2}).C2\overline{\mathrm{A}}_{get}G)$$

.4) The interpreter C3 controls the goal delegate manager C4 to read the goal database G, according to goal G and current cognitive of K2, then adopts rhombus thinking, decomposes goal G forms goal decomposition tree. Goal delegation may produce new internal event E. Exteics semantics is as below:

$$C3\vec{A}_{control}(C4\overleftarrow{A}_{read}\,\mathrm{G}.\mathrm{C4}\overleftarrow{\mathrm{A}}_{r\,\mathrm{hom}bus}(\mathrm{g},\mathrm{K2}).$$

$$C4\overline{\mathrm{A}}_{get}SGi,(\mathrm{i}=1,2,...,\mathrm{n})\,|\,\mathrm{SGi}\,\vec{\mathrm{A}}_{\mathrm{producte}}E)$$

5) The interpreter C3 controls service manager C5 to read goal tree, according current cognitive to find active services Si to achieve each leaf sub goal. Service aggregation may produce internal event E. Extenics semantics is as below:

$$C3\vec{A}_{control}(C5\overleftarrow{A}_{read}\,\mathrm{SGi}.\mathrm{C5}\vec{\mathrm{A}}_{find}\,\mathrm{Si}.$$

$$C5\overline{\mathrm{A}}_{assemble}Si,(\mathrm{i}=1,2,...,\mathrm{n})\,|\,\mathrm{Si}\,\vec{\mathrm{A}}_{\mathrm{producte}}E)$$

6) The interpreter C3 Controls plan manager C6 to read service set, according order to encapsulate all service into a plan P. Plan dynamic scheduling may produce internal event E. Extenics semantic is as below:

$$C3\vec{A}_{control}(C6\overleftarrow{A}_{read}\,\mathrm{Si}.\mathrm{C6}\vec{\mathrm{A}}_{order}\,\mathrm{Si}.$$

$$C6\overline{\mathrm{A}}_{\mathrm{encapsulate}}(Si,\mathrm{P}),(\mathrm{i}=1,2,...,\mathrm{n})\,|\,P\vec{A}_{producte}E)$$

7) The interpreter C3 controls executor C7 to read the plan library P, according plan sequence to execute each plan Pi. Extenics semantics is as below:

$$C3\vec{A}_{control}(C7\overleftarrow{A}_{read}\,\mathrm{P}.\mathrm{C7}\vec{\mathrm{A}}_{execute}\,\mathrm{Pi},\mathrm{i}=1,2,...,\mathrm{n})$$

8) The executor C7 outputs results through the interface I4 to impact environment Env, Env will through the input interface I3 to form new event queue E. Extenics semantics is as below:

$$\mathrm{C7}\vec{\mathrm{A}}_{execute}(\mathrm{Pi},\mathrm{i}=1,2,...,\mathrm{n}).\mathrm{C7}\vec{\mathrm{A}}_{\mathrm{output}}$$

$$\mathrm{I4}\vec{A}_{output}Env\vec{A}_{input}I3\vec{A}_{input}E$$

9) The interpreter C3 executes 3) again, so again and again, Extenics self-adaptive cycle is formed.

7 We can use of extension analysis reduce to vary base-element nets. Considering goal is decomposed into sub goals can constitute a basic-element implication tree which contains sub goals, leaves sub goal and implication services. The implication tree express hierarchy implication relation between goals and services, and three types of services which achieve the leaves sub goal are composition service (Com-Service), provide services (Pro-Service) and request service (Req-Service). Figure 5 shows the implication tree of goal decomposition and service implementation.

In figure 5, the goal is decomposed into several sub goal SGi, the optimization sub goal SGi by rhombus reasoning, then released to the goal database G. In the service container S, provide service PS active aggregate for achieving sub goal SGi. Some sub goal SGi cannot be realized, the request services RS are published to find and invoke other intelligent service component ISCs related provide services Si. Extenics semantics is as below:

$$Goal\,\vec{A}_{Decomposition}\,\mathrm{SGi}\,\vec{A}_{r\,\mathrm{hom}bus}\,\mathrm{SGi}\,\overleftarrow{\mathrm{A}}_{\mathrm{implement}}(Si\,\overleftarrow{A}_{assemble}PS$$

$$|\,Si\,\overleftarrow{A}_{requment}\,\mathrm{RS}\,\overleftarrow{\mathrm{A}}_{\mathrm{provide}}\,\mathrm{ISCs}),(\mathrm{i}=1,2,...,\mathrm{n})$$

$$Goal \Leftarrow \begin{cases} SubGoal1 \Leftarrow \begin{cases} SubGoal11 \Leftarrow Pro-Service1 \\ SubGoal12 \Leftarrow Com-Service2 \Leftarrow \begin{cases} Pro-Service2 \\ Pro-Service3 \\ Req-Service1 \end{cases} \\ SubGoal13 \Leftarrow SubGoal131 \Leftarrow Pro-Service5 \\ \cdots\cdots \\ SubGoal1n \Leftarrow Pro-Service6 \end{cases} \\ SubGoal2 \Leftarrow \begin{cases} SubGoal21 \Leftarrow Pro-Service7 \\ SubGoal22 \Leftarrow Pro-Service8 \\ SubGoal23 \Leftarrow Pro-Service9 \\ \cdots\cdots \\ SubGoal2m \Leftarrow Req-Service2 \end{cases} \\ \cdots\cdots \\ SubGoal\,n \Leftarrow \begin{cases} SubGoaln1 \Leftarrow Pro-Service10 \\ SubGoaln2 \Leftarrow Com-Service2 \Leftarrow \begin{cases} Pro-Service12 \\ Pro-Service13 \\ Pro-Service14 \end{cases} \\ SubGoaln3 \Leftarrow SubGoal131 \Leftarrow Pro-Service11 \\ \cdots\cdots \\ SubGoalnn \Leftarrow Req-Service3 \end{cases} \end{cases}$$

Figure 5. Goal Tree & Services Implementation

8 The extension transformation for dynamic evolution

Using dynamic base-element net for expressing self-adaptive software may be convenient for using the extension transformation to analyze and study the dynamic evolution of self-adaptive software. For example, the base-element net shown in Figure 4, if a new user requirements arise, intelligent service component perceives the new event, and generate a new goal, then decompose into new sub goals and new leave sub goals, and news service for leave sub goals have to been found. For example, to add a new sub goal and new service, i.e., transformation T to Goal, get a new goal Goal':

$$TGoal = Goal' = Goal \cup$$

$$(SG_{New}\,\overleftarrow{\mathrm{A}}_{\mathrm{implement}}S_{New}\,\overleftarrow{A}_{requment}\,\mathrm{RS}\,\overleftarrow{\mathrm{A}}_{\mathrm{provide}}\,\mathrm{ISCs})$$

We find that the dynamic evolution of intelligent service component can be represented by Extenics conduction transforms chain:

$$TEnv \rightarrow TEvent \rightarrow TGoal \rightarrow TSubGoal \rightarrow$$

$$TService \rightarrow TPlan \rightarrow TAction \rightarrow TEnv$$

This conduction transform chain conforms to self-adaptive software Monitoring-Analysis-Planning- Executing (MAPE) cycle chain.

9 The dependent function for self-adaptability evaluating

The dependent function includes elementary, simple and discrete dependent function etc. The details can refer to reference [3]. For example, to evaluate the intelligence service component self-adaptability in response to environment changes, the key is to determine the number of m=l+r services that can realize n leave goals, l is local service number of the intelligent service component, r is

remote service number of other intelligent service component. The discrete dependent function is as below:

$$k(x)=\begin{cases}1, n << l\\ 0.75, n < l\\ 0.5, n = l\\ 0.25, l < n < m\\ 0, n = m, l = 0\\ -0.25, n \geq m, l = 0\\ -0.5, n > m, l = 0\\ -1, n >> m = 0\end{cases} \quad (1)$$

When local services number l are very much leave goals number n, the dependent function value is 1; when local services number l are more leave goals number n, the dependent function value is 0.75; when local services number l are equal leave goals number n, the dependent function value is 0.5; when local services number l less than leave goals number n, but remote service number r and local service number l is greater than leave goals number n, the dependent function value is 0.25; when local services number are 0, and remote service number r equals leave goals number n, the dependent function value is 0; when local services number l are 0, and remote service number r is more than or equal to leave goals number n, the dependent function value is -0.25; when local services number l are 0, and remote service number r is less than or equal to leave goals number n, the dependent function value is -0.5; when local services number l are 0, and remote service number r very less than leave goals number n, the dependent function value is -1.

## III. CONCLUSION

By Introducing Extenics theory and methods, this paper studies self-adaptive software base-element net method, and use the extensible analysis methods to enrich evolution knowledge of intelligent service component. We discuss self-adaptive mechanism description, and analyze influence degree, trend about the extension transformation act on intelligent service component. The dependent function gives quantitative evaluation of self-adaptive software. Further work is to compare and analyze of the Extenics formal methods with other formal methods for expending application scope of Extenics formal method.

## REFERENCES

[1] W. Cai, Extension set and non compatible problems. *Chinese Journal of scientific exploration*, 1, pp.83-97, 1983.

[2] W. Cai, C.Y. Yang, Basic theory and methodology on Extenics. *Chinese Science Bulletin*, 58 (13), pp.1190-1199, 2013. https://doi.org/10.1360/972012-1472

[3] C.Y. Yang, W. Cai, *Extenics: Theory, Method and Application*, Science Press: Beijing, 2013.

[4] C.Y. Yang, X, S, Li, Research Progress in Extension Innovation Method and its Applications. *Industrial Engineering Journal*, 15(1), pp.131-137, 2012.

[5] C.Y. Yang, W. Cai, A formalized system of extension information- knowledge- intelligence. *CAAI Transactions on Intelligent Systems*, 2(3), pp.9-11, 2007.

[6] F.Q. Yang, H. Mei, Internetware: new forms of software of the future. *China education network*, pp.52-54, 2005.

[7] K.Q. He, R. Peng, W. Liu, J. Wang, B. Li, *Networked Software*, Science Press: Beijing, 2008.

[8] I. Jorstad, S. Dustdar, D.V. Thanh, A service oriented architecture framework for collaborative services. *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises ( WET ICE 2005)*, pp.121-125, 2005. https://doi.org/10.1109/wetice.2005.11

[9] X.W. Gu, Z.D. Lu, A Pi-calculus Based Formal Model for BPEL4WS Web Service Composition. C*omputer Science*, 30(3), pp.69-74, 2007.

[10] Jorg P. Muller, Massimo Cossentino, *Agent-Oriented Software Engineering XIII:* 13TH International Workshop, AOSE 2012, Valencia, Spain, June 4, 2012, Revised Selected Papers. Lecture Notes in Computer Science, Vol. 7852, Springer 2013.

[11] Z.Z. Shi, J.H. Zhang, J.P. Yue, X. Yang. A Cognitive Model for Multi-Agent Collaboration. *International Journal of Intelligence Science*, 4, pp.1-6, 2014. https://doi.org/10.4236/ijis.2014.41001

[12] R. Fan, Modeling Extenics Innovation Software by Intelligent Service Components. *The Open Cybernetics & Systemics Journal*, 8, pp.1-7, 2014. https://doi.org/10.2174/1874110X01408010001

## AUTHORS

**R. Fan** is with the Guangdong Ocean University, Zhanjiang, 524088 China (fanrui@ gdou.edu.cn).

**Y. F. Chen** is with the Guangdong Ocean University, Zhanjiang,524088 China (yuefengch71@126.com).

**S. C. Cao** is with the University of Illinois at Urbana-Champaign, Champaign, IL 61801 USA, (author@nrim.go.jp).

**G. B. Lei** is with the Guangdong Ocean University, Zhanjiang, 524088 China (718686316@qq.com).

**C. Yue** is with the Guangdong Ocean University, Zhanjiang, 524088 China (273901837@qq.com).

This work was supported in part by the scientific and technological projects of Guangdong province, China under Contract 2014A040402010. Submitted, 26 October 2016. Published as resubmitted by the authors on 27 November 2016.