Research Methodology on Code Clone Detection with Refactoring Using Textual and Metrics Analysis in Software

Dr.G.Anil Kumar

Sr. Assistant Professor CSE MGIT Hyderabad T.S. India

Abstract *The act of cut, copy and pasting code fragments and making minor, non–functional alterations, is a major problem for large, industrial software systems. It leads to duplicated code fragments known as code clones. The major consequence of cloning in the code is that it makes the maintenance process difficult. Finding out the reused fragment of code in any application is usually called as code clone detection. In the process of software maintenance, evolution clones are considered to be harmful because it increases the complexity of the system. From the evolution of the clone detection, it provides improved results and decreases the complexity of the system for better maintenance.*

Through the state of art in code cloning, one can understand clone detection process is mainly focused on detection of a line after line or detection based on tokenization. This technique makes the system complex and takes long time to process the source code to find the clones in it. If a code fragment is not an exact copy, but the functionality shows that it is similar to another code fragment, then current clone detection system unable to find out such type of clones.

The proposed research model for detection of clone approach shows that the detection process is easier and it has produced efficient results. This approach is a process of combining textual approach and metric analysis of the given source code for detection of all four types of clones presented in a given set of code fragment in java source code. All the detected clone pairs are grouped together to form clone clusters and they are stored in files. All the detected clones can be automatically refactored if it is required by the programmer.

Different semantics have been formulated and the values of these semantics have been used in the process of clone detection. These metrics along with textual analysis provide a very less complexity in figuring out the clones and provide accurate results.

Efficiency of the technique is measured in terms of Precision and Recall values. The results of the proposed method are compared with the bench mark tools like Clone DR, CCFinder and other techniques. The analysis of the experimental results shows that Precision and Recall values are improved and they are better than the previous techniques.

of Social networking and wide range of smart devices and Internet applications has lead to creation of extremely large sets of complex data referred as BIGDATA. IOT is interconnection of physical things using intelligent devices like sensors etc. and operating them with ease[1].

Keywords — *Code Clone, Clone detection, refactoring, metrics, textual analysis,*

I. INTRODUCTION, OVERVIEW, CONCLUSION OF RESEARCH WORK AND FUTURE ENHANCEMENTS

The act of cut, copy and pasting code fragments and making minor, non–functional alterations, is a major problem for large, industrial software systems. It leads to duplicated code fragments known as code clones. The major consequence of cloning in the code is that it makes the maintenance process difficult. Finding out the reused fragment of code in any application is usually called as code clone detection. In the process of software maintenance, evolution clones are considered to be harmful because it increases the complexity of the system. From the evolution of the clone detection, it provides improved results and decreases the complexity of the system for better maintenance.

Through the state of art in code cloning, one can understand clone detection process is mainly focused on detection of a line after line or detection based on tokenization. This technique makes the system complex and takes long time to process the source code to find the clones in it. If a code fragment is not an exact copy, but the functionality shows that it is similar to another code fragment, then current clone detection system unable to find out such type of clones.

The proposed research model for detection of clone approach shows that the detection process is easier and it has produced efficient results. This approach is a process of combining textual approach and metric analysis of the given source code for detection of all four types of clones presented in a given set of code fragment in java source code. All the detected clone pairs are grouped together to form clone clusters and they are stored in files. All the detected clones can be

automatically refactored if it is required by the programmer.

Different semantics have been formulated and the values of these semantics have been used in the process of clone detection. These metrics along with textual analysis provide a very less complexity in figuring out the clones and provide accurate results.

Efficiency of the technique is measured in terms of Precision and Recall values. The results of the proposed method are compared with the bench mark tools like Clone DR, CCFinder and other techniques. The analysis of the experimental results shows that Precision and Recall values are improved and they are better than the previous techniques. CONCLUSIONS

Present Work:

Software maintenance is a very important phase of the development life cycle. Reducing the software maintenance overhead is an activity that makes the software industry more comfortable. When it is compared with any other product the software product customers expect more of maintenance of the product because they feel it is flexible (i.e. it is simply writing few instructions). So, changes can be accommodated easily at any time. But software engineering literature says it is a myth.

Software clone is more dangerous in large software systems. Most of the times cloning happens due to copy and paste activity only. Almost every developer thinks to save developing time, so he uses this activity because developing code from the scratch takes more time. Sometimes time constraints force developers to turn towards cloning. Some maintenance engineers accidentally produce these clones. Although it seems to be an effective and simple solution to the developer's problems, usually these cloning activities are documented and it leads to number of negative effects on the quality of the software. It increases the total number of lines of code of the system and lines of code that needs to be maintained.

Clone detection is an ongoing research area and the existing literature is overwhelmed in detecting and eliminating clones from software systems. The literature presented in literature survey topic of this research work gave several dimensions of code cloning. Many existing methods and tools have been compared and discussed. It is very important to identify the clones present in the code, at the same time there should be some solution proposed to this problem. The existing refactoring methods can give solutions to the code cloning problems. Refactoring has been used effectively in the proposed method.

In this work, a light-weight method has been proposed to identify functional clones. This method uses the computation of several metrics in combination with simple textual analysis technique. The usage of metrics with existing exponential rate of comparison overhead of the other methods is reduced to minimum number of comparisons. This is possible by early analysis of potential clones and applying comparisons only on code fragments that are identified as clones in this analysis. Since the string matching/textual comparison is performed over the short listed candidates, a higher amount of recall could be obtained.

The Proposed work is divided into two stages. The first one is selection of potential clones and the second one is comparison of potential clones. The proposed technique detects exact clones on the basis of metric match and then by text match. Potential clones are compared line-by-line to determine whether two potential clones really are clones of each other. The experiments proved that this method can do better than existing systems in finding and classifying the clones in JAVA. The Precision and Recall values that are obtained describe the efficiency of the work proposed. It has been proved that Precision 98% and Recall 96% is achievable in code cloning. In addition it also identifies the functional clones.

Future Enhancements

Though the proposed technique is working efficiently for Programming languages like JAVA, it can be extended to find clones in multiple languages. When it comes to identify only type I, type II and type III clones this method can identify clones in almost all object oriented programming languages. Research work can be extended not only to find the clones but also to remove the actual clones. Though refactoring process has been used, it can be fully automated and implemented so that no human intervention is required.

The proposed method is experimented on medium sized software applications only. These applications are of 10 to 15 KLOC only. Experiments on large scale systems can be conducted to observe efficiency of the method. The parameters for the efficiency are taken only in the form of precision and recall values. It also can be extended to scalability, portability and robustness etc.

II. APPENDIX

Programs of the case study are presented earlier.. This is a class which is taken from a package where only few methods were present. These classes are taken because the methods which are present in these classes are useful to demonstrate the code duplication.

Program 1

```
import java.util.*;

import java.lang.*;
```

```
public class TestFileOne{

        intp,q=1,r;

        double VALUE; /* the number which we
                         need to find factrorial */

public int factorial(int n)  {  /*  factorial
function
    using recursive function */

    if(n == 0)   {

     return 1;

    }  else     {

     return n * factorial(n-1);
                          }
        }

    public intgcdOne(int a, int b) {
     while (b != 0)                    {
      if (a > b)
        {
       a = a - b;
      } else
        {
       b = b - a;
        }
     }
     return a;
    }

    public intmul(int a, int b){

     int n = 0, p=0;

     p=p+1;

     for(int i = 0; i < b; i++)        {

      n += a;
     }

     return n;
    }
        public int factorial1 ( int VALUE ){ /*
factorial using for loop */

        for (p=1; p<=VALUE; p++)

        q = q*p;
        return q;
        }
}
```

This program is similar to the program.1 with few changes in it

Program 2

```
import java.util.*;

import java.lang.*;


public class TestFileTwo {

    public int factorial2 (int n){ /* factorial using
recursive function */
     if(n == 0)   {
      return 1;
                          }        else      {
      return n * factorial2(n-1);
     }
    }

    public intgcdTwo(int c, int d) {
     while (d != 0)          {
      if (c > d)              {
       c = c - d;
        } else    {
       d = d - c;
        }
     }
     return c;
    }

    public double mulTwo(double a, long b)
        {
     double n = 0.0;
     for(long i = 0l; i < b; i++)
      n += a;

     return n;
    }
}
```

REFERENCES

[1] IEEE Standards for Software Maintenance, IEEE Standard 1219, 1998.

[2] ISO/IEC. Software Engineering – SoftwareMaintenance.ISO/IEC 14764,1999.

[3] L. Arthur. Software Evolution: The Software Maintenance Challenge. Wiley,1988.

[4] S.W.L.Yip and T. Lam,"A software maintenance survey", In Proc. of the 1stAsia-Pacific Software Engineering Conference, pp 70–79, Dec 1994.

[5] S. Chidamber and C. Kemerer "A metric suite for object-oriented design" IEEE Transactions on Software Engineering, 25(5):476–493, Jun 1994.

[6] Jennie Brown, MattiTeinonen "Software configuration management A ClearCase for IBM Rational" IBM Redbooks, 2004

[7] Robert Tairas, "Clone detection and refactoring", Proceeding of OOPSLA '06 Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications, pp. 780-781, New York, USA, 2006

[8] Chanchal K. Roy, James R. Cordya and Rainer Koschkeb, "Comparison and Evaluation of Code Clone Detection

Techniques and Tools: A Qualitative Approach", Journal Science of Computer Programming, Vol. 74, No.7, pp. 470-495, May 2009.

[9] Ira D. Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant Anna and Lorraine Bier, "Clone Detection Using Abstract Syntax Trees", Proceedings of the International Conference on Software Maintenance, pp. 368, Washington DC, USA 1998

[10] Douglas Martin, James R. Cordy, "Analyzing Web Service Similarity Using Contextual Clones", ACM Journal, 2011.

[11] M.Fowlor, "Refactoring: improving the design of existing code", Addison Wesley, 1999.

[12] William C Wake, "Refactoring work book", Pearson Education Inc, 2004.

[13] Magiel Bruntink, Arie van Deursen,Remco van Engelen, and Tom Tourwe, "On the Use of Clone Detection for Identifying Crosscutting Concern Code", IEEE Transactions On Software Engineering, Vol. 31, No. 10, pp. 804-818, October 2005

[14] Abouelhoda M.I., Kurtz S.andOhlebusch E, "The enhanced suffix array and its applications to genome analysis", In Proc. Workshop on Algorithms in Bioinformatics, vol.2452, pp. 449–463, Berlin, 2002

[15] Hamid Abdul Basit and Stan Jarzabek, "Detecting Higher-level Similarity Patterns in Programs", European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 1-10 Lisbon, Sept. 2005

[16] Lingxiao Jiang, Zhendong Su and Edwin Chiu, "Context-based detection of clone-related bugs", Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, pp. 55 – 64, New York, USA, 2007.

[17] Chanchal Kumar Roy and James R Cordy, "A Survey on Software Clone Detection Research", Computer and Information Science, Vol. 115, No. 541, pp. 115, 2007

[18] J Howard Johnson "Identifying Redundancy in Source Code Using Fingerprints" In Proceeding of the 1993 Conference of the Centre for Advanced Studies Conference (CASCON'93), pp. 171-183, Toronto, Canada, October 1993.

[19] Zhenmin Li, Shan Lu, SuvdaMyagmar, and Yuanyuan Zhou,"CP-Miner: Finding Copy-Paste and Related Bugs in Large-Scale Software Code", In IEEE Transactions on Software Engineering, Vol. 32(3): 176-192, March 2006.

[20] Christopher Brown, Simon Thompson. "Clone Detection and Elimination for Haskell", ACM journal, 2010.

[21] Elizabeth Burd and Malcolm Munro "Investigating the maintenance implications of the replication of code" in Proceedings of the 13th International Conference on Software Maintenance (ICSM'97), Bari, Italy, September 1997.

[22] Matthias Rieger. "Effective Clone Detection without Language Barriers" Ph.D. Thesis, University of Bern, Switzerland, June 2005.

[23] Magiel Bruntink, "Aspect Mining using Clone Class Metrics", In Proceedings of the 1stWorkshop on Aspect Reverse Engineering, 2004.

[24] Fabio Calefato, FilippoLanubile, Teresa Mallardo, "Function Clone Detection in Web Applications: A Semi automated Approach", Journal of Web Engineering, Vol. 3, No.1, pp.003-021, 2004.

[25] Andrew Walenstein and Arun Lakhotia, "The Software Similarity Problem in Mal-ware Analysis", In Proceedings Dagstuhl Seminar 06301: Duplication, Redundancy, and Similarity in Software, pp.10, Dagstuhl, Germany, July 2006.

[26] Giuliano Antoniol, Gerardo Casazza, Massimiliano Di Penta, and Ettore Merlo,"ModelingClones Evolution through Time Series", In Proceedings of the 17th IEEE International Conference on Software Maintenance (ICSM'01), pp. 273-280, Florence, Italy, November 2001.

[27] W-K. Chen, B. Li, and R. Gupta, "Code Compaction of Matching Single-Entry MultipleExit Regions", In Proceedings of the 10th Annual International Static Analysis Symposium (SAS'03), pp. 401-417, San Diego, CA, USA, June 2003.

[28] Magiel Bruntink, Arie van Deursen, TomTourwe and Remco van Engele, "An Evaluation of Clone Detection Techniques for Identifying Crosscutting Concerns", In Proceedings of the 20th IEEE International Conference on Software Maintenance, pp. 200- 209, Washington DC, USA 2004.

[29] Ira D. Baxter and Dale Churchett, "Using Clone Detection to Manage a Product Line", proceedings of International conference on Clone detection using abstract syntax trees, pp. 1-3,1998.

[30] Heejung Kimy, Yungbum Jungy, Sunghun Kimx and Kwangkeun Yi, "MeCC: Memory Comparison-based Clone Detector", 33rd International Conference on Software Engineering, Waikiki, Honolulu, Hawaii, May 21-28, 2011

[31] Florian Deissenboeck, Benjamin Hummel, Elmar Jurgens, Bernhard Schatz, Stefan Wagner, Jean-François Girard and Stefan Teucher, "Clone detection in automotive model-based development", Proceedings of the 30th international conference on Software engineering, pp. 613-622,New York, NY, USA,2008

[32] Robert Tairas, Jeff Gray and Ira Baxter, "Visualization of clone detection results", In Proceedings of the 2006 OOPSLA Workshop on Eclipse Technology Exchange ACM, pp. 50-54, New York, USA, 2006.

[33] Minhaz F.Zibran and ChanchalK.Roy, "Towards Flexible Code Clone Detection, Management, and Refactoring in IDE", Fifth International Workshop on Software Clones, Waikiki, Hawaii, USA, May 23, 2011

[34] M. Kim, L. Bergman, T.A. Lau, and D. Notkin, "An Ethnographic Study of Copy and Paste Programming Practices in OOPL," In Proceedings of International Symposium on Empirical Software Eng. (ISESE '04), pp. 83-92, Aug. 2004

[35] M. Rieger, S. Ducasse, and G. Golomingi, "Tool Support for Refactoring Duplicated OO Code," In Proceedings of European Conference on Object- Oriented Programming (ECOOP '99), pp. 177-178, June 1999.

[36] Rainer Koschke, RaimarFalke and Pierre Frenzel, "Clone Detection Using Abstract Syntax Suffix Trees," In Proc. of the 13th Working Conference on Reverse Engineering, Benevento, pp. 253 - 262, Oct 2006.

[37] Stephane Ducasse, Oscar Nierstrasz and Matthias Rieger, "Research On the effectiveness of clone detection by string matching," Journal of Software Maintenance and Evolution: Research and Practice, Vol. 18, No. 1, pp. 37-58, 2006.

[38] Michael Toomim, Andrew Begel and Susan L. Graham, "Managing Duplicated Code with Linked Editing", In the Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing(VL/HCC'04), pp. 173-180, Rome, Italy, September 2004.

[39] YueJia, David Binkley, Mark Harman, Jens Krinke and Makoto Matsushita, "KClone: A Proposed Approach to Fast Precise Code Clone Detection," In Proceedings of the Third International Workshop on Detection of Software Clones (IWSC 2009), pp. 12-16, 2009.

[40] R. R. Brooks, P. Y. Govindaraju, M. Pirretti, N. Vijaykrishnan and M. Kandemir, "Clone Detection in Sensor Networks with Ad Hoc and Grid Topologies," International Journal of Distributed Sensor Networks, Vol. 5, pp. 209–223, 2009.

[41] Shinji Kawaguchi, TakanobuYamashinay, HidetakeUwanoz, KyhoheiFushida, Yasutaka Kamei, MasatakaNagura and Hajimu Iida, "SHINOBI: A Tool for Automatic Code Clone Detection in the IDE," In Proceedings of the 16th Working Conference on Reverse Engineering, pp. 313 - 314, Oct 2009.

[42] Nam H. Pham, HoanAnh Nguyen, Tung Thanh Nguyen, Jafar M. Al-Kofahi and Tien N. Nguyen, "Complete and Accurate Clone Detection in Graph-based Models," In Proceedings of the 31st International Conference on Software Engineering, Washington, DC, 2009.

[43] Kodhai. E, Kanmani. S, Kamatchi. A, Radhika.R and VijayaSaranya. B, "Detection of Type-1 and Type-2 Code Clones Using Textual Analysis and Metrics," In Proceedings of the 2010 International Conference on Recent Trends in

Information, Telecommunication and Computing, Washington, DC, pp. 241-243, 2010.
[44] ArmijnHemel, Karl TrygveKalleberg, Rob Vermaas, and EelcoDolstrac, "Finding Software License Violations Through Binary Code Clone Detection," In Proceedings of the 8th working conference on Mining software repositories, New York, NY, May 2011.
[45] Kodhai.E, Perumal.A, and Kanmani.S, "Clone Detection using Textual and Metric Analysis to figure out all Types of Clones," In Proceedings of the International Joint Journal Conference on Engineering and Technology (IJJCET 2010), pp. 99 - 103, 2010.
[46] F.Calefato, F.Lanubile and T.Mallardo, "Function Clone Detection in WebApplications: A Semiautomated Approach," Journal of Web Engineering, Vol.3, No. 1, pp. 3–21, 2004.
[47] T.Kamiya, S.Kusumoto, and K.Inoue, "CCFinder: A MultiLinguistic Token-Based Code Clone Detection System for Large Scale Source Code," IEEE Trans. Software Eng., Vol. 28, No. 7, pp. 654-670, July 2002.
[48] B.Baker, "On Finding Duplication and Near-Duplication in Large SoftwareSystems", In Proceedings of the Second Working Conference on Reverse Engineering (WCRE'95), pp. 86–95, Toronto, Ontario, Canada, July 1995.
[49] Ettore Merlo1, "Detection of Plagiarism in University Projects Using Metrics based Spectral Similarity," In the Dagstuhl Seminar: Duplication, Redundancy, and Similarity in Software, 2007.
[50] S.Ducasse, M.Rieger and S.Demeyer, "A Language Independent Approach for Detecting Duplicated Code," In Proceedings of the 15th International Conference on Software Maintenance (ICSM'99), pp. 109–118, September 1999.
[51] [52] Richard Wettel, RaduMarinescu "Archeology of Code Duplication: Recovering Duplication Chains From Small Duplication Fragments", In Proceedings of the 7th Inter-national Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05), pp.8, Timisoara, Romania, September 2005.
[52] Andrian Marcus and Jonathan I. Maletic,"Identification of high-level concept clones in source code", In Proceedings of the 16th IEEE International Conference on Automated Software Engineering (ASE'01), pp. 107-114, San Diego, CA, USA, November 2001.
[53] B.Baker, "Finding Clones with Dup: Analysis of an Experiment," IEEETransactions on Software Engineering, Vol. 33, No. 9, pp. 608–621, 2007.
[54] J.R. Cordy and C.K. Roy, "The NiCad Clone Detector," In 19th International Conference on Program Comprehension, Kingston, Canada, June 2011.
[55] Aoun Raza, Gunther Vogel, Erhard Plaodereder,"Bauhaus: A Tool Suite for Program Analysis and Reverse Engineering", In Proceedings of the 11th Ada-Europe International Conference on Reliable Software Technologies, LNCS 4006, pp. 71-82, Porto, Portugal, June 2006.
[56] W.Yang, "Identifying Syntactic Differences Between Two Programs," Software Practice and Experience, Vol. 21, No. 7, pp. 739–755, July 1991.
[57] V. Wahler, D. Seipel, J. Gudenberg and G. Fischer, "Clone Detection in Source Code by Frequent Itemset Techniques" In Proceedings of the 4th IEEE International Workshop Source Code Analysis and Manipulation (SCAM), pp.128–135, Chicago, IL, USA, September 2004.
[58] Williams Evans, and Christopher Fraser, "Clone Detection via Structural Abstraction", In the Proceedings of the 14th Conference on Reverse Engineering (WCRE'07), Vancouver, BC, Canada, October 2007.
[59] Robert Tairas, Jeff Gray, "Phoenix-Based Clone Detection Using Suffix Trees", In the Proceedings of 44th annual Southeast regional conference (ACM-SE'06), pp. 679-684, Melbourne, Florida, USA, March 2006.
[60] J.Mayrand, C.Leblanc and E.Merlo, "Experiment on the Automatic Detection ofFunction Clones in a Software System Using Metrics," In the Proceedings of the 12th International Conference on Software Maintenance (ICSM'96), pp. 244–253, Monterey, CA, USA, November 1996.
[61] UdiManber, "Finding similar files in a large file system" In Proceedings of the Winter1994 Usenix Technical Conference, pp. 110, San Francisco, USA,January 1994.
[62] Seunghak Lee, IryoungJeong,"SDD: High Performance Code CloneDetection Systemfor Large Scale Source Code", In Proceedings of the Object Oriented Programming Systems Languages and Applications Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOP-SLA Companion'05), pp. 140-141, San Diego, CA, USA, October 2005.
[63] Neil Davey, Paul Barson, Simon Field, Ray J Frank, "The Development of a Software Clone Detector", International Journal of Applied Software Technology, Vol. 1, pp. 219-236, 1995
[64] B. BoUobas, "Random Graphs", Cambridge University Press, 2001.
[65] Mohammed Abdul Bari and Dr. Shahanawaj Ahamad, "Code Cloning: The Analysis, Detection and Removal", International Journal of Computer Applications (0975 – 8887) Volume 20, No.7, April 2011
[66] Khurram Zeeshan Haider, Tabassam Nawaz, Sami ud Din, and Ali Javed, "Efficient Source Code Plagiarism Identification Based on Greedy String Tilling", International Journal of Computer Science and Network Security, Vol.10 No.12, December 2010.
[67] Florian Deissenboeck, Benjamin Hummel, Elmar Jurgens, Stefan Wagner, "Do Code Clones Matter", In the Proceedings of the 31st international conference on Software engineering, pp. 485-495,New York, NY, USA,2009.
[68] Bettenburg, Nicola, Weyi Shang , Ibrahim, and W.Adams "An Empirical Study on Inconsistent Changes to Code Clones at Release Level", Conference on Reverse Engineering, 2009.
[69] Suresh Thummalapenta, Luigi Cerulo, LerinaAversano, Massimiliano Di Penta, "An empirical study on the maintenance of source code clones", Empirical Software Engineering, February, Volume 15, Issue 1, pp. 1-34, 2010.
[70] Foyzur Rahman, Christian Bird, PremkumarDevanbu, "Clones: what is that smell?", Empirical Software Engineering, August, Vol. 17, Issue 4-5, pp. 503-530, 2012.
[71] Jeremy R. Patel, Robert Tairas and Nicholas A. Kraft, "Clone evolution: a systematic review", Journal of Software: Evolution and Process, Volume 25, Issue 3, pp. 261–283, March 2013.
[72] Florian Deissenboeck , Benjamin Hummel, Elmar Juergens, Michael Pfaehler, Bernhard Schaetz, "Model clone detection in practice", In the Proceedings of the 4th International Workshop on Software Clones, pp. 57-64, 2010.
[73] Robert Tibshirani, Pei Wang, "Spatial smoothing and hot spot detection for CGH data using the fused lasso", Biostatistics, pp.1-7, 2007.
[74] HiranDhanjia, Michel Doumitha, Olivier Clermontb, Erick Denamurb, Russell Hopea, David M. Livermorea, Neil Woodforda, "Real-time PCR for detection of the O25b-ST131 clone of Escherichia coli and its CTX-M-15-like extended-spectrum lactamases", International Journal of Antimicrobial Agents, Vol.36, pp.355–358, 2010.