

Automatic Generation of Abstract Views for Legacy Software Comprehension

Philippe Dugerdil, Julien Repond

Dept. of Information Systems

HEG-Univ. of Applied Sciences of Western Switzerland

7 route de Drize, CH-1227 Geneva, Switzerland

+41 22 388 17 00

philippe.dugerdil@hesge.ch, julien.repond@hesge.ch

ABSTRACT

One of the main motivations for the reverse engineering of software programs is to help with software comprehension. Although several techniques have been presented in the literature to reverse-architect software, the corresponding views usually do not help much. In fact, most of the published techniques recover the architecture of the software by focusing on the abstract properties of the components such as coupling and coherence. We claim that the recovered components should rather represent abstract functional entities whose behavior could be understood independently from the others. Then, an abstract view of the system would represent the interactions between such functional entities. In this paper we present a technique and a tool able to generate abstract sequence diagrams to represent the global working of legacy programs. This shows the main interactions between abstract functional components. When comparing the automatically generated sequence diagrams to the one a developer would produce by hand, we realized that the representation were very close. Our work could then be considered as a first step to the automatic generation of human-understandable abstract views of the working of legacy programs.

Categories and Subject Descriptors

D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement - *Restructuring, reverse engineering, and reengineering.*

General Terms: Design, Experimentation, Algorithm.

Keywords

Reverse-engineering, functional component, software clustering, dynamic analysis, abstract view, sequence diagram.

1. INTRODUCTION

For the last 15 years, legacy software reverse-engineering has become an important field of computer science and an active field of research. It is a known fact that software maintenance is the most expensive activity over the software life cycle [20]. In these costs, the task of software understanding takes the lion's share.

Nowadays a tremendous research activity is spent on techniques to help with software understanding, often targeted at generating views of software system architecture [10]. To give a precise meaning to the word “understanding” in the context of reverse-engineering, we borrow the definition by Biggerstaff et al. [8]: “A person understands a program when able to explain the program, its structure, its behavior, its effects on its operational context, and its relationships to its application domain in terms that are qualitatively different from the tokens used to construct the source code of the program”. Moreover, it is known for long that to “understand” a large software system, which is a critical task in reverse-engineering, its structural aspects (i.e. its architecture) are more important than any single algorithmic component [29]. But the problem with legacy software is that, often, the only reliable source of information is its mere source code which hardly contains clues on higher level components [7]. Therefore it is hard to create representations of the architecture of the legacy software from its mere source code. Besides, we know that software architecture can be represented through many views [10], each targeting a particular purpose.

In this work, we target the construction of an architectural view that could help with understanding the code. We have called it the *functional architecture view* of the system, i.e. the structure of *functional components* and their relationships which implement the high level business function of the software. In this context, a *functional component* is a cohesive set of classes implementing some relevant processing step during the execution of a given scenario. To identify the functional components of a legacy software, we compute clusters of dynamically-correlated classes from the post mortem execution trace of the system. Class clustering has been around in software reverse-engineering for quite some time. Basically, it could be based on static (i.e. source code) or dynamic (i.e. execution trace) analysis techniques or both. For example, Tzerpos and Holt [32] use the identification of patterns of structure in the source code to cluster classes. The latter approach rests on the hypothesis that comprehension is enhanced when familiar structures are discovered in the code. We follow the same initial goal: the identification of clusters that would help with software comprehension. However we are not interested by the “discovery” of structural patterns in the code but by functional clusters of strongly interacting classes. Then we developed a tool to display the functional components on top of the actual structure of the code. Often, these components will span several packages. This is the first step in our reverse-engineering technique. Since we target software understanding, we must show how the functional components (i.e. dynamic clusters) interact to implement the business scenarios (instances of the use-cases) of the system. Then, we developed a tool to automatically generate the sequence diagram of cluster interactions, to show the role they played in the actual working of the legacy system. The idea of generating a sequence

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISEC'10, February 25-27, 2010, Mysore, India.

Copyright 2010 ACM 978-1-60558-922-0/10/02...\$10.00.

diagram from the execution trace of a system is not new. However in any industrial size system, the execution trace usually contains on the order 10^5 to 10^7 events (method calls). Therefore the raw presentation of the corresponding sequence diagram is unusable. A solution would be to work on sophisticated sequence diagram viewers equipped with zooming and navigation features to limit the quantity of information to display at any single time. This is the approach taken by Bennett et al. [6]. But this approach does not help much with the task of separating out important information from the “noise”. In our approach, the information to be displayed is limited in principle by the very clustering technique: all interactions internal to functional components are not represented. As a result we obtain a very readable sequence diagram that shows the key interaction relevant to the execution of the scenario. This representation seems to be quite close to what the developer would actually have produced by hand while explaining his system.

2. FUNCTIONAL ARCHITECTURE

- The architecture of the system has not necessarily been design with the goal to help understanding. In particular it could have been design to satisfy any one of the required quality attribute [4].
- The maintenances on the software system are likely to have spoiled its original architecture over time [5].

- The resulting set of functional components will usually span several packages (or any other static grouping of classes or elements).
- Two functional components may share some of their classes.

In a sense, a functional component can be seen as a grouping of classes' responsibilities involved in some step of the processing associated to a scenario. Therefore, if the responsibilities of a class are used in several steps of a given scenario we may well end up with two clusters (functional component) sharing the same class. This is again why packages and dynamic cluster (functional components) are not the same concept. This is illustrated in figure 1. The grayed zone represents a functional component (cluster) grouping the responsibilities involved at a given moment in time while executing a scenario.

Figure 1: Packages, classes and clusters

Once we have recovered the use-cases, we must record the working of the system i.e. its execution trace (sequence of method invocations). This is obtained by instrumenting the source code of the legacy system. The inserted statements in the source code will generate information about the executed function in some standard format that can be analyzed later to identify clusters and represent them on different architectural views. The instrumentation is performed using our own tool developed using JavaCC. The steps are the following. First, the legacy source code is parsed to get its syntax tree. Next, the latter is explored to attach the relevant instrumentation code. The resulting code is recompiled and executed following the chosen scenario.

sets of classes that are tightly coupled [21]. While most of the published work rest on the static analysis of the code, we focus on dynamic techniques i.e. on execution trace analysis and trace segmentation. Some researchers have tried to analyze the execution trace to identify regular collaboration patterns among classes [16]. However implementation details, in particular the use of polymorphism, could lead to many collaboration variants that should nonetheless be considered the same as far as collaboration is concerned. Moreover, another problem is to decide on the maximum “distance” between two classes in the call sequence to still consider them as close collaborators. In summary if one takes into account all the variants of the interaction patterns and the problem of the distance between the classes to build the clustering algorithm, we end up with a computationally intensive algorithm, similar to finding a subgraph of a graph which is known to be NP complete. Moreover it is important to realize that execution traces of all but trivial programs are generally very large (on the order of $10^5 \sim 10^7$ events). For example, in one of our experiments, the legacy system generated an execution trace with more than 7 millions of events (method calls). A true simplification of the problem arises if we make the hypothesis that the classes which collaborate heavily occur closely in the execution trace. Therefore, an easy way to check for collaborating classes is to split the execution trace in contiguous segments and observe the class presence in each segment (figure 2). In this figure, each vertical string represents a method call. The result of the analysis of the occurrences of each class is presented as a binary occurrence vector for the class. If a class occurs at least once in a segment, the corresponding element in the occurrence vector will be 1, 0 otherwise. As an example the presence of Class1 has been highlighted in the figure, and its corresponding occurrence vector V_{c1} is presented below.

The classes will then be clusterized according to the distance between their occurrence vectors. This is the principle of the technique we proposed to identify the collaborating application classes and that has been presented elsewhere [11][12]. Since we focus on application classes i.e. the classes that the developers of the system wrote themselves, the instrumentation will only target these classes. Therefore the resulting execution trace does not contain calls to the system classes or the classes pertaining to some framework such as Hibernate [18] or Spring [19]. This greatly reduces the number of events to process. To refer to a common terminology in software clustering techniques [3], the *entities* to cluster are the classes and the single *formal feature* used to group classes is the binary occurrence of the classes in each segments. Therefore we compute for each class the feature vector which represents the presence (1) of absence (0) of the class in each segment. This feature, that represents a *sibling link* between the classes, is a kind of shortcut to actually represent a *direct link* [3].

package1 class1 method1	package2 class2 method2	package3 class3 method3	package4 class4 method4	package5 class5 method5	package6 class6 method6	package7 class7 method7	package8 class8 method8	package9 class9 method9	package10 class10 method10	package11 class11 method11	package12 class12 method12	package13 class13 method13	package14 class14 method14	package15 class15 method15	package16 class16 method16	package17 class17 method17	package18 class18 method18	package19 class19 method19	package20 class20 method20	package21 class21 method21	package22 class22 method22	package23 class23 method23	package24 class24 method24	package25 class25 method25	package26 class26 method26	package27 class27 method27	package28 class28 method28	package29 class29 method29	package30 class30 method30	package31 class31 method31	package32 class32 method32	package33 class33 method33	package34 class34 method34	package35 class35 method35	package36 class36 method36	package37 class37 method37	package38 class38 method38	package39 class39 method39	package40 class40 method40	package41 class41 method41	package42 class42 method42	package43 class43 method43	package44 class44 method44	package45 class45 method45	package46 class46 method46	package47 class47 method47	package48 class48 method48	package49 class49 method49	package50 class50 method50	package51 class51 method51	package52 class52 method52	package53 class53 method53	package54 class54 method54	package55 class55 method55	package56 class56 method56	package57 class57 method57	package58 class58 method58	package59 class59 method59	package60 class60 method60	package61 class61 method61	package62 class62 method62	package63 class63 method63	package64 class64 method64	package65 class65 method65	package66 class66 method66	package67 class67 method67	package68 class68 method68	package69 class69 method69	package70 class70 method70	package71 class71 method71	package72 class72 method72	package73 class73 method73	package74 class74 method74	package75 class75 method75	package76 class76 method76	package77 class77 method77	package78 class78 method78	package79 class79 method79	package80 class80 method80	package81 class81 method81	package82 class82 method82	package83 class83 method83	package84 class84 method84	package85 class85 method85	package86 class86 method86	package87 class87 method87	package88 class88 method88	package89 class89 method89	package90 class90 method90	package91 class91 method91	package92 class92 method92	package93 class93 method93	package94 class94 method94	package95 class95 method95	package96 class96 method96	package97 class97 method97	package98 class98 method98	package99 class99 method99	package100 class100 method100	package101 class101 method101	package102 class102 method102	package103 class103 method103	package104 class104 method104	package105 class105 method105	package106 class106 method106	package107 class107 method107	package108 class108 method108	package109 class109 method109	package110 class110 method110	package111 class111 method111	package112 class112 method112	package113 class113 method113	package114 class114 method114	package115 class115 method115	package116 class116 method116	package117 class117 method117	package118 class118 method118	package119 class119 method119	package120 class120 method120	package121 class121 method121	package122 class122 method122	package123 class123 method123	package124 class124 method124	package125 class125 method125	package126 class126 method126	package127 class127 method127	package128 class128 method128	package129 class129 method129	package130 class130 method130	package131 class131 method131	package132 class132 method132	package133 class133 method133	package134 class134 method134	package135 class135 method135	package136 class136 method136	package137 class137 method137	package138 class138 method138	package139 class139 method139	package140 class140 method140	package141 class141 method141	package142 class142 method142	package143 class143 method143	package144 class144 method144	package145 class145 method145	package146 class146 method146	package147 class147 method147	package148 class148 method148	package149 class149 method149	package150 class150 method150	package151 class151 method151	package152 class152 method152	package153 class153 method153	package154 class154 method154	package155 class155 method155	package156 class156 method156	package157 class157 method157	package158 class158 method158	package159 class159 method159	package160 class160 method160	package161 class161 method161	package162 class162 method162	package163 class163 method163	package164 class164 method164	package165 class165 method165	package166 class166 method166	package167 class167 method167	package168 class168 method168	package169 class169 method169	package170 class170 method170	package171 class171 method171	package172 class172 method172	package173 class173 method173	package174 class174 method174	package175 class175 method175	package176 class176 method176	package177 class177 method177	package178 class178 method178	package179 class179 method179	package180 class180 method180	package181 class181 method181	package182 class182 method182	package183 class183 method183	package184 class184 method184	package185 class185 method185	package186 class186 method186	package187 class187 method187	package188 class188 method188	package189 class189 method189	package190 class190 method190	package191 class191 method191	package192 class192 method192	package193 class193 method193	package194 class194 method194	package195 class195 method195	package196 class196 method196	package197 class197 method197	package198 class198 method198	package199 class199 method199	package200 class200 method200	package201 class201 method201	package202 class202 method202	package203 class203 method203	package204 class204 method204	package205 class205 method205	package206 class206 method206	package207 class207 method207	package208 class208 method208	package209 class209 method209	package210 class210 method210	package211 class211 method211	package212 class212 method212	package213 class213 method213	package214 class214 method214	package215 class215 method215	package216 class216 method216	package217 class217 method217	package218 class218 method218	package219 class219 method219	package220 class220 method220	package221 class221 method221	package222 class222 method222	package223 class223 method223	package224 class224 method224	package225 class225 method225	package226 class226 method226	package227 class227 method227	package228 class228 method228	package229 class229 method229	package230 class230 method230	package231 class231 method231	package232 class232 method232	package233 class233 method233	package234 class234 method234	package235 class235 method235	package236 class236 method236	package237 class237 method237	package238 class238 method238	package239 class239 method239	package240 class240 method240	package241 class241 method241	package242 class242 method242	package243 class243 method243	package244 class244 method244	package245 class245 method245	package246 class246 method246	package247 class247 method247	package248 class248 method248	package249 class249 method249	package250 class250 method250	package251 class251 method251	package252 class252 method252	package253 class253 method253	package254 class254 method254	package255 class255 method255	package256 class256 method256	package257 class257 method257	package258 class258 method258	package259 class259 method259	package260 class260 method260	package261 class261 method261	package262 class262 method262	package263 class263 method263	package264 class264 method264	package265 class265 method265	package266 class266 method266	package267 class267 method267	package268 class268 method268	package269 class269 method269	package270 class270 method270	package271 class271 method271	package272 class272 method272	package273 class273 method273	package274 class274 method274	package275 class275 method275	package276 class276 method276	package277 class277 method277	package278 class278 method278	package279 class279 method279	package280 class280 method280	package281 class281 method281	package282 class282 method282	package283 class283 method283	package284 class284 method284	package285 class285 method285	package286 class286 method286	package287 class287 method287	package288 class288 method288	package289 class289 method289	package290 class290 method290	package291 class291 method291	package292 class292 method292	package293 class293 method293	package294 class294 method294	package295 class295 method295	package296 class296 method296	package297 class297 method297	package298 class298 method298	package299 class299 method299	package300 class300 method300	package301 class301 method301	package302 class302 method302	package303 class303 method303	package304 class304 method304	package305 class305 method305	package306 class306 method306	package307 class307 method307	package308 class308 method308	package309 class309 method309	package310 class310 method310	package311 class311 method311	package312 class312 method312	package313 class313 method313	package314 class314 method314	package315 class315 method315	package316 class316 method316	package317 class317 method317	package318 class318 method318	package319 class319 method319	package320 class320 method320	package321 class321 method321	package322 class322 method322	package323 class323 method323	package324 class324 method324	package325 class325 method325	package326 class326 method326	package327 class327 method327	package328 class328 method328	package329 class329 method329	package330 class330 method330	package331 class331 method331	package332 class332 method332	package333 class333 method333	package334 class334 method334	package335 class335 method335	package336 class336 method336	package337 class337 method337	package338 class338 method338	package339 class339 method339	package340 class340 method340	package341 class341 method341	package342 class342 method342	package343 class343 method343	package344 class344 method344	package345 class345 method345	package346 class346 method346	package347 class347 method347	package348 class348 method348	package349 class349 method349	package350 class350 method350	package351 class351 method351	package352 class352 method352	package353 class353 method353	package354 class354 method354	package355 class355 method355	package356 class356 method356	package357 class357 method357	package358 class358 method358	package359 class359 method359	package360 class360 method360	package361 class361 method361	package362 class362 method362	package363 class363 method363	package364 class364 method364	package365 class365 method365	package366 class366 method366	package367 class367 method367	package368 class368 method368	package369 class369 method369	package370 class370 method370	package371 class371 method371	package372 class372 method372	package373 class373 method373	package374 class374 method374	package375 class375 method375	package376 class376 method376	package377 class377 method377	package378 class378 method378	package379 class379 method379	package380 class380 method380	package381 class381 method381	package382 class382 method382	package383 class383 method383	package384 class384 method384	package385 class385 method385	package386 class386 method386	package387 class387 method387	package388 class388 method388	package389 class389 method389	package390 class390 method390	package391 class391 method391	package392 class392 method392	package393 class393 method393	package394 class394 method394	package395 class395 method395	package396 class396 method396	package397 class397 method397	package398 class398 method398	package399 class399 method399	package400 class400 method400	package401 class401 method401	package402 class402 method402	package403 class403 method403	package404 class404 method404	package405 class405 method405	package406 class406 method406	package407 class407 method407	package408 class408 method408	package409 class409 method409	package410 class410 method410	package411 class411 method411	package412 class412 method412	package413 class413 method413	package414 class414 method414	package415 class415 method415	package416 class416 method416	package417 class417 method417	package418 class418 method418	package419 class419 method419	package420 class420 method420	package421 class421 method421	package422 class422 method422	package423 class423 method423	package424 class424 method424	package425 class425 method425	package426 class426 method426	package427 class427 method427	package428 class428 method428	package429 class429 method429	package430 class430 method430	package431 class431 method431	package432 class432 method432	package433 class433 method433	package434 class434 method434	package435 class435 method435	package436 class436 method436	package437 class437 method437	package438 class438 method438	package439 class439 method439	package440 class440 method440	package441 class441 method441	package442 class442 method442	package443 class443 method443	package444 class444 method444	package445 class445 method445	package446 class446 method446	package447 class447 method447	package448 class448 method448	package449 class449 method449	package450 class450 method450	package451 class451 method451	package452 class452 method452	package453 class453 method453	package454 class454 method454	package455 class455 method455	package456 class456 method456	package457 class457 method457	package458 class458 method458	package459 class459 method459	package460 class460 method460	package461 class461 method461	package462 class462 method462	package463 class463 method463	package464 class464 method464	package465 class465 method465	package466 class466 method466	package467 class467 method467	package468 class468 method468	package469 class469 method469	package470 class470 method470	package471 class471 method471	package472 class472 method472	package473 class473 method473	package474 class474 method474	package475 class475 method475	package476 class476 method476	package477 class477 method477	package478 class478 method478	package479 class479 method479	package480 class480 method480	package481 class481 method481	package482 class482 method482	package483 class483 method483	package484 class484 method484	package485 class485 method485	package486 class486 method486	package487 class487 method487	package488 class488 method488	package489 class489 method489	package490 class490 method490	package491 class491 method491	package492 class492 method492	package493 class493 method493	package494 class494 method494	package495 class495 method495	package496 class496 method496	package497 class497 method497	package498 class498 method498	package499 class499 method499	package500 class500 method500	package501 class501 method501	package502 class502 method502	package503 class503 method503	package504 class504 method504	package505 class505 method505	package506 class506 method506	package507 class507 method507	package508 class508 method508	package509 class509 method509	package510 class510 method510	package511 class511 method511	package512 class512 method512	package513 class513 method513	package514 class514 method514	package515 class515 method515	package516 class516 method516	package517 class517 method517	package518 class518 method518	package519 class519 method519	package520 class520 method520	package521 class521 method521	package522 class522 method522	package523 class523 method523	package524 class524 method524	package525 class525 method525	package526 class526 method526	package527 class527 method527	package528 class528 method528	package529 class529 method529	package530 class530 method530	package531 class531 method531	package532 class532 method532	package533 class533 method533	package534 class534 method534	package535 class535 method535	package536 class536 method536	package537 class537 method537	package538 class538 method538	package539 class539 method539	package540 class540 method540	package541 class541 method541	package542 class542 method542	package543 class543 method543	package544 class544 method544	package545 class545 method545	package546 class546 method546	package547 class547 method547	package548 class548 method548	package549 class549 method549	package550 class550 method550	package551 class551 method551	package552 class552 method552	package553 class553 method553	package554 class554 method554	package555 class555 method555	package556 class556 method556	package557 class557 method557	package558 class558 method558	package559 class559 method559	package560 class560 method560	package561 class561 method561	package562 class562 method562	package563 class563 method563	package564 class564 method564	package565 class565 method565	package566 class566 method566	package567 class567 method567	package568 class568 method568	package569 class569 method569	package570 class570 method570	package571 class571 method571	package572 class572 method572	package573 class573 method573	package574 class574 method574	package575 class575 method575	package576 class576 method576	package577 class577 method577	package578 class578 method578	package579 class579 method579	package580 class580 method580	package581 class581 method581	package582 class582 method582	package583 class583 method583	package584 class584 method584	package585 class585 method585	package586 class586 method586	package587 class587 method587	package588 class588 method588	package589 class589 method589	package590 class590 method590	package591 class591 method591	package592 class592 method592	package593 class593 method593	package594 class594 method594	package595 class595 method595	package596 class596 method596	package597 class597 method597	package598 class598 method598	package599 class599 method599	package600 class600 method600	package601 class601 method601	package602 class602 method602	package603 class603 method603	package604 class604 method604	package605 class605 method605	package606 class606 method606	package607 class607 method607	package608 class608 method608	package609 class609 method609	package610 class610 method610	package611 class611 method611	package612 class612 method612	package613 class613 method613	package614 class614 method614	package615 class615 method615	package616 class616 method616	package617 class617 method617	package618 class618 method618	package619 class619 method619	package620 class620 method620	package621 class621 method621	package622 class622 method622	package623 class623 method623	package624 class624 method624	package625 class625 method625	package626 class626 method626	package627 class627 method627	package628 class628 method628	package629 class629 method629	package630 class630 method630	package631 class631 method631	package632 class632 method632	package633 class633 method633	package634 class634 method634	package635 class635 method635	package636 class636 method636	package637 class637 method637	package638 class638 method638	package639 class639 method639	package640 class640 method640	package641 class641 method641	package642 class642 method642	package643 class643 method643	package644 class644 method644	package645 class645 method645	package646 class646 method646	package647 class647 method647	package648 class648 method648	package649 class649 method649	package650 class650 method650	package651 class651 method651	package652 class652 method652	package653 class653 method653	package654 class654 method654	package655 class655 method655	package656 class656 method656	package657 class657 method657	package658 class658 method658	package659 class659 method659	package660 class660 method660	package661 class661 method661	package662 class662 method662	package663 class663 method663	package664 class664 method664	package665 class665 method665	package666 class666 method666	package667 class667 method667	package668 class668 method668	package669 class669 method669	package670 class670 method670	package671 class671 method671	package672 class672 method672	package673 class673 method673	package674 class674 method674	package675 class675 method675	package676 class676 method676	package677 class677 method677	package678 class678 method678	package679 class679 method679	package680 class680 method680	package681 class681 method681	package682 class682 method682	package683 class683 method683	package684 class684 method684	package685 class685 method685	package686 class686 method686	package687 class687 method687	package688 class688 method688	package689 class689 method689	package690 class690 method690	package691 class691 method691	package692 class692 method692	package693 class693 method693	package694 class694 method694	package695 class695 method695	package696 class696 method696	package697 class697 method697	package698 class698 method698	package699 class699 method699	package700 class700 method700	package701 class701 method701	package702 class702 method702	package703 class703 method703	package704 class704 method704	package705 class705 method705	package706 class706 method706	package707 class707 method707	package708 class708 method708	package709 class709 method709	package710 class710 method710	package711 class711 method711	package712 class712 method712	package713 class713 method713	package714 class714 method714	package715 class715 method715	package716 class716 method716	package717 class717 method717	package718 class718 method718	package719 class719 method719	package720 class720 method720	package721 class721 method721	package722 class722 method722	package723 class723 method723	package724 class724 method724	package725 class725 method725	package726 class726 method726	package727 class727 method727	package728 class728 method728	package729 class729 method729	package730 class730 method730	package731 class731 method731	package732 class732 method732	package733 class733 method733	package734 class734 method734	package735 class735 method735	package736 class736 method736	package737 class737 method737	package738 class738 method738	package739 class739 method739	package740 class740 method740	package741 class741 method741	package742 class742 method742	package743 class743 method743	package744 class744 method744	package745 class745 method745	package746 class746 method746	package747 class747 method747	package74
-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-----------

argument based on implementation details is sensible, it remains to be demonstrated that the classes identified using the utilityhood metrics [15] do identify implementation-only classes and do not contain functional code. In fact, their algorithm is based on the fan-in metrics i.e. the number of static links referencing a given class. Then a ratio is computed between the fan-in and the total number of classes in the system (or component, depending on the scope of the search). All classes whose utilityhood value is bigger than a predefined threshold are considered utilities. However, nothing prevents a class that contains a utility method to contain also some functional code. Besides, the compression ratio using these techniques is not enough to get a resulting execution trace of a displayable size. In a case study, Hamou-Lhadj and Lethbridge removed utility classes calls as well as other implementation details such as methods of inner classes, accessors, constructors/finalizers and the like [15]. The compression ratio was about 30 i.e. there were 30 times less events in the compressed trace than in the original trace. Such compression factor would compress a trace of 6.10^5 events (that we got in the first case study) to about 20.10^3 events. This is still much too big to be displayed as a sequence diagram. It is important to note that the compression technique based on utilities produces some noise reduction but does not address class clustering. Other published compression techniques include the pattern-based trace sequence summarization [16]. There, recurring patterns of calls are identified and written only once in the resulting trace. Later occurrences of the same pattern refer to the first occurrence with the number of repetitions. To identify repetitions, Hamou-Lhadj and Lethbridge applied the common subexpression algorithm borrowed from DNA sequencing techniques [16]. The problem with this approach is that the resulting trace format is difficult to represent as a sequence diagram since there are references and loopback in the compressed format. Moreover, in their case studies Hamou-Lhadj and Lethbridge found compression factor between 11 and 40, which is still insufficient if the target is to display a trace of a million events.

As far as clustering is concerned, the algorithms can be divided in two broad categories: partitional and hierarchical [21]. In architecture recovery, most of the techniques pertain to the hierarchical category because partitional algorithms must start with an initial set of clusters that is supposed to be known in advance, which is usually not the case in architecture recovery. On the other hand, hierarchical algorithm start with the basic elements (for example the classes) and, using some distance metric, try to agglomerate them in clusters. Then, at each step of the algorithm, a distance between each unclustered element and the clusters is computed and the element is associated with the closest cluster. Therefore, noise reduction is important in the agglomerative hierarchical clustering since, depending on the distance metrics used, otherwise unrelated classes may be clustered together, especially if the SLA (Single Linkage Algorithm [21]) clustering algorithm is used. For example, if utility class A would be strongly linked to class B and C, even if B and C are unrelated, the three classes would be part of the same cluster because A would “glue” B and C.

Our approach based on graph-theoretic concepts does not need a preprocessing step of noise reduction. Since the goal is to help with the understanding of legacy code, which is supposed to be badly structured, we do not want to make the hypothesis that some class is an implementation-only class and does not contains functional code. Therefore all classes in the system are considered when computing class clusters. But the very algorithm of maximal clique requires all classes to be strongly linked. Then, even if utility class A would be strongly linked to B and C, these two

classes would never be part of the same cluster if they are not closely collaborating. Once the clusters are computed, we display their interactions in a sequence diagram. Since the clustering is based on heavy class interactions, i.e. lots of events, there will be much less events to display if we remove the inner working of the clusters. Figure 4 shows the principle of trace reduction based on clusters. Since (A,B) and (C,D) form two clusters, we only display the interactions between the classes that are member of different clusters. In this case we would then have two interacting functional components.

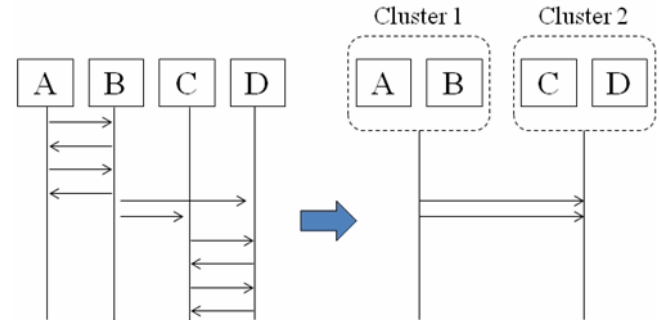


Figure 4: Cluster sequence diagram

5. FUNCTIONAL ARCHITECTURE VIEW

The best support for legacy software understanding is to generate relevant views of the system’s architecture. Then, we designed a new architectural view that we called the *Functional Architecture View* of a system with two representations:

1. The functional components (clusters) displayed on top of the package structure of a program.;
2. The interaction of the functional components (clusters) in a sequence diagram.

This new view has been integrated in the IBM’s Rational Software Architect (RSA) environment as an Eclipse plugin. The latter computes the clusters and displays the two representations using the tools available in RSA. With these views, a maintenance engineer could highlight the classes of the legacy system that work heavily together to implement some business function and observe their distribution among the packages. Then he could display the way they interact as functional components. The Functional Architecture view was first tested by hand and its relevance assessed with respect to the understanding of the code [12]. Then we decided to automate this technique using the plugin technology of Eclipse [25]. In figure 5 we present the functional components part of the view. In the top pane we see the architecture of the packages and the classes as reconstructed by hand using the Java to UML transformation feature of RSA. In the bottom pane, we see a new tab in called “Cluster View” which displays the list of clusters (functional components) identified in the execution trace. Then, a color can be assigned to each cluster and the classes will be colored according to the cluster they belong to in diagram on the top. With this representation, the maintenance engineer can observe the span of the functional components over the actual static structure of the code.

For a sequence diagram to be useful for system understanding, it must be of a limited size (a few dozen of events at most). Even if the clustering of classes drastically reduces the number of events to be considered, the remaining number of event in industrial size systems is generally still too big to be readable by humans.



Figure 5: Functional components

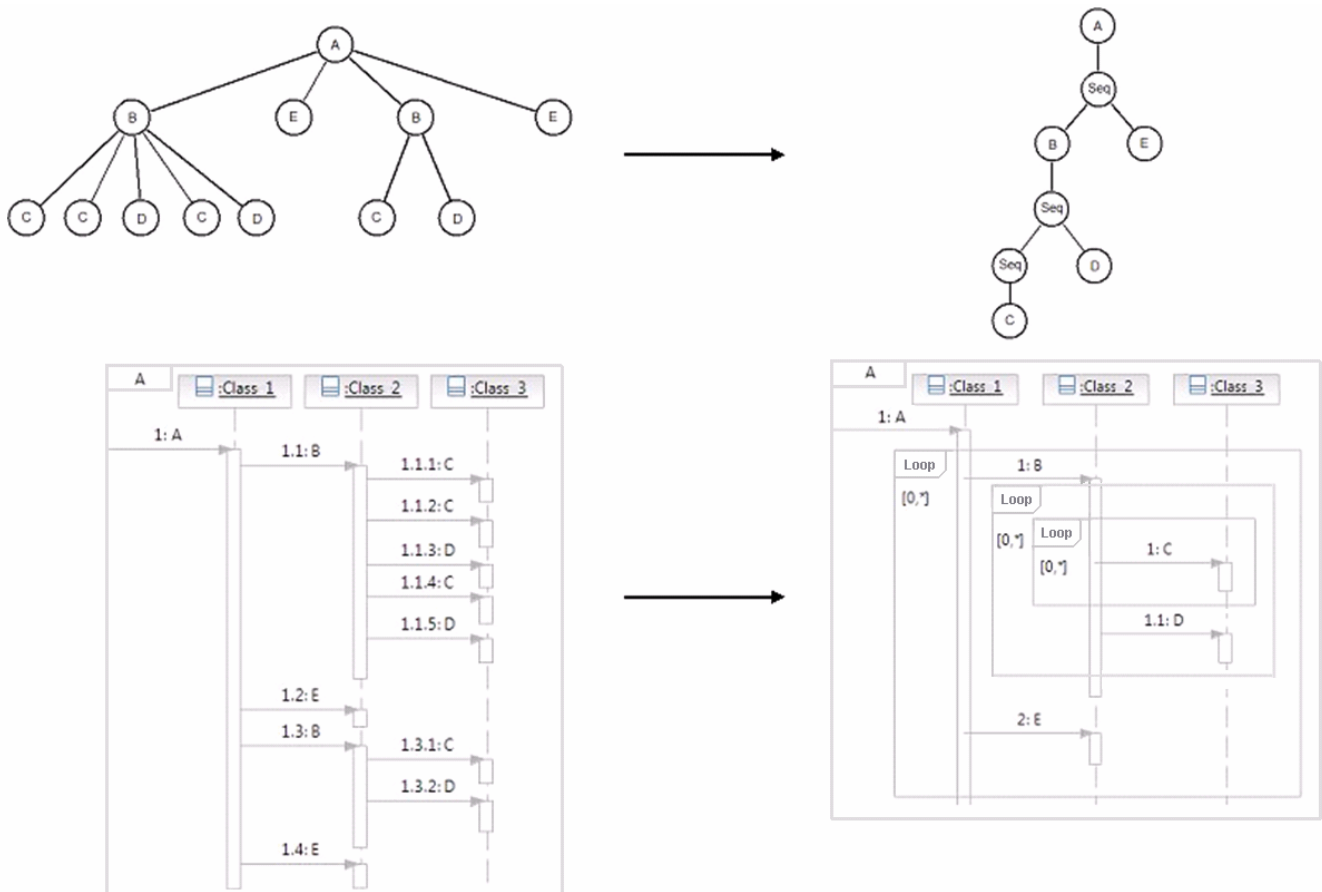


Figure 6: Trace repetitions compression and corresponding UML2 sequence diagrams

Therefore, we applied two extra event compression techniques:

- Removal of accessor methods;
- Compression of contiguous repetitions of similar patterns of events.

The algorithm used to detect event patterns repetitions is similar to what Hamou-Lhadj and Lethbridge proposed to produce the Compact Trace Format (CTF) [14]. This works bottom up (from the leaves to the root). At each level, the contiguous repetition of a pattern in the call tree of an execution trace is detected and replaced by a new SEQ node whose sons are the elements of the pattern. This indicates that the elements are repeated many times. Once a level of nodes is “compressed” we move up to the next level to find new repetitions and so on. This is shown in figure 6. In the left part we present the original call tree and the corresponding sequence diagram. On the right part we present the “compressed” version and the associated sequence diagram. For example, among the sons of the “B” node on the left we find a contiguous repetition of “C” nodes. This becomes a new SEQ node with “C” as its single son. Since a single occurrence of a node can be considered a unique repetition, we discover that the sons of “B” are now: (“sequence of C”, “D”), (“sequence of C”, “D”). We can then replace it with a new SEQ node for the pattern: (“sequence of C”, “D”), as shown in the final call tree on the right. Again, since a single occurrence of a node is matchable against any number of repetitions of the same node, the same compression applies to the sons of “A”. The translation to UML2 sequence diagram is easy: the SEQ node becomes a loop fragment and its sons become the sequence that is repeated. However, there is still a problem to solve. As explained in paragraph 3, our clustering algorithm may well generate overlapping clusters. This is when a subset of classes is strongly associated to two different clusters. But the two clusters cannot be merged because the other classes of both clusters are not coupled enough to each other. This is shown in figure 7. However nothing prevents a class outside the two clusters to call the methods of the classes that are located at the intersection of the clusters. This is shown in figure 8 where the class C0 calls a method in cluster A (C1) as well as a method at the intersection of both clusters (C3).

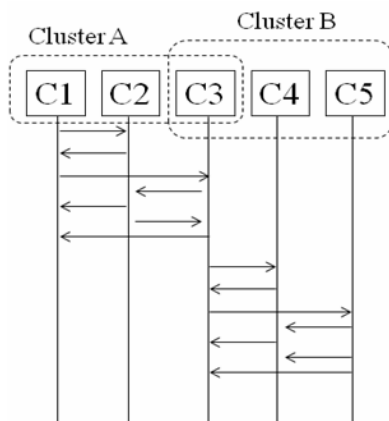


Figure 7: Overlapping clusters

But the latter creates a representation problem: how to display a method call to 2 different entities at the same time in a sequence diagram? To solve this problem we created the concept of “union-cluster” to represent an entity whose contents belong to several

clusters (functional component). Therefore, in a sequence diagram, we would represent both the individual clusters and the union clusters. To visually identify the clusters and union clusters in UML2 diagrams, we designed two new stereotypes stored as a new profile. They are represented in figure 9 where the containment relationship is represented using the aggregation association.

Figure 10 presents a sequence diagram displaying the calls to the clusters A and B and to the union cluster between clusters A and B. However, there remains a problem to be solved. What to do with the classes that are not clustered? Due to the fact that our clustering technique only groups closely interacting classes, there will remain application classes that are not member of any cluster [13]. However our goal is to identify and represent the functional components that are involved in the implementation of the main steps of a scenario. Therefore the interactions between the unclustered classes are not relevant at this granularity level. These classes are then grouped in a single container just to show when ones of these classes interacts with the functional components. This is represented by yet another cluster that we called “Undefined”. We could see an example of this in the figures 12 and 13 of the case study.

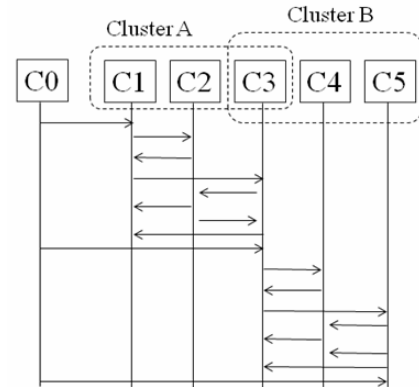


Figure 8: Calls to the class at the intersection of two clusters

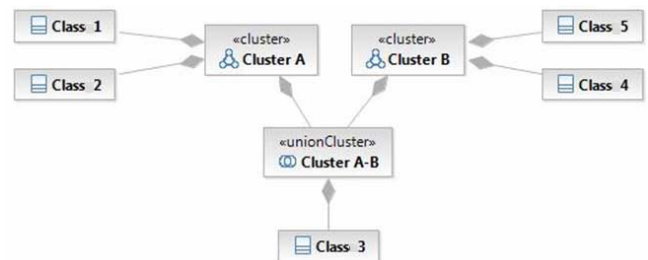


Figure 9: Cluster and “union cluster” stereotype

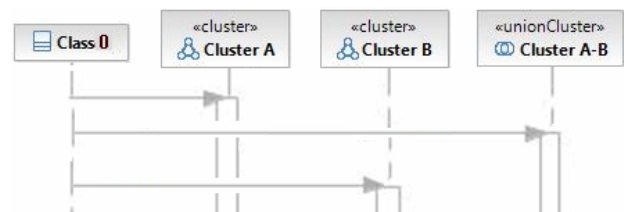


Figure 10: Sequence diagram with a “union cluster”

As for the Eclipse plugin, figure 11 presents its main screen in the IBM's Rational Software Architect environment. This works as follow. The first step of a reverse engineering experiment is to create a new UML modeling project. Then we generate the UML class diagram of the legacy Java system using the standard Java to UML transformation of RSA. The resulting model is then stored in the new project. In parallel, the source code of the legacy system is instrumented, recompiled, executed and the execution trace generated. The latter can now be loaded and analyzed using the trace analysis features of our plugin (whose screen is displayed in figure 11).

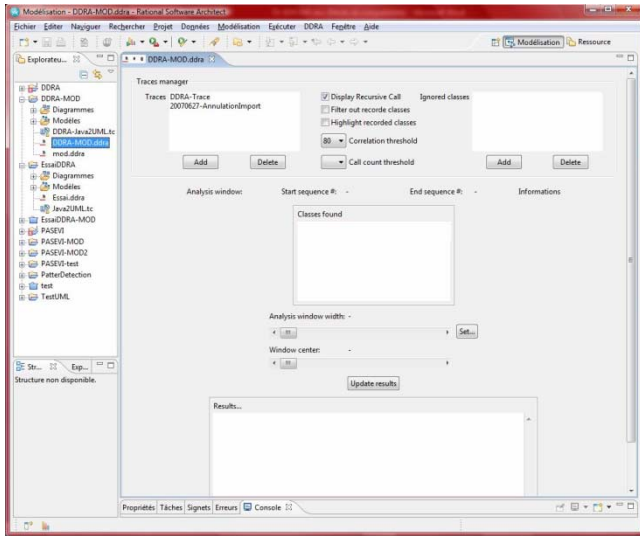


Figure 11: The plugin's GUI

Once this analysis is completed, the identified clusters are displayed in the project explorer together with the classes they contain as showed in figure 12. Finally we can generate a sequence diagram that would only take into account the discovered clusters as well as the cluster "Undefined" that gathers the remaining, unclustered, classes.

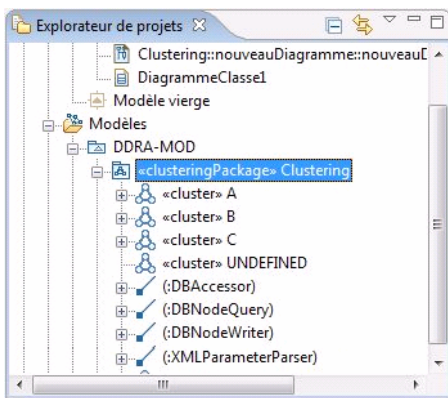


Figure 12: Project explorer with clusters

6. CASE STUDY

Our tool and technique have been applied to the trace analyser itself that we instrumented before running a standard cluster analysis scenario. The execution trace had 365'000 events that we segmented in 600 contiguous segments. The exact number of

segment must be computed with respect to the number of classes in the execution trace as explained in [12]. A common factor is 32 (i.e. the number of segments must be 32 times the number of classes in the execution trace). In this example, the clustering algorithm found 3 clusters in the system.

Compression	Remaining events
Clustering	324
Clustering + Accessors removal	166
Clustering + Accessors removal + Repetitive sequences compression	45

Figure 13: Trace compression results

In figure 13, we present the compression factors of the trace using the three steps presented above: the clustering, the removal of accessors and the compression of repetitive sequences of events. Altogether the clustering of the trace and the extra compression steps took about 15 seconds to perform on a standard PC (3Mhz, 3Gb RAM). The clustering alone shows an impressive compression factor of 1000. If one adds the two other compression techniques one gets a compression factor of 8000: from the starting trace of 365'000 events one ends up with only 45 events which is quite manageable.

The question now is: are the remaining 45 events be of any help to understand the working of the system? As expected we found that the remaining event were quite informative and expressed clearly the main steps in the computation of the clusters.

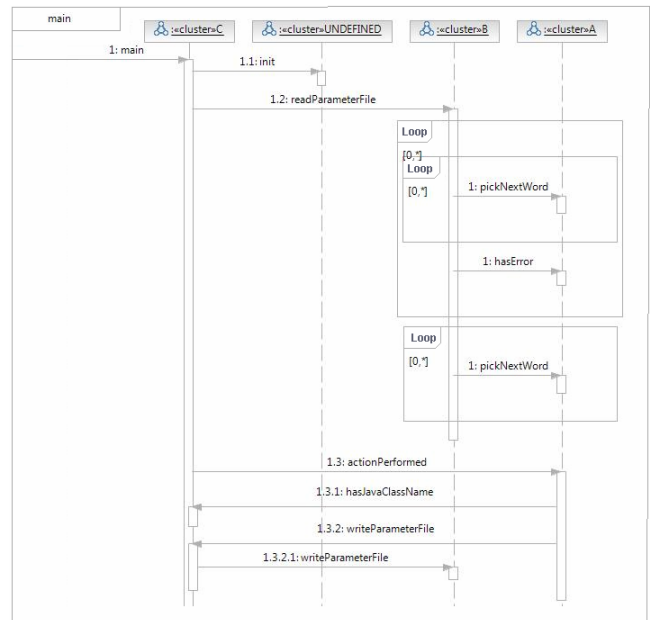


Figure 14: Sequence diagram for the initialization step

For example, figure 14 presents the part of the sequence diagram that represents the initialization of the work. First, all the processing parameters are read from a parameter file. As we can see, the main method of the parameter file reading step are represented in the diagram, especially the "readParameterFile"

event and the loop to read the keywords in the file. The “writeParameterFile” event at the end represents the recording of the chosen parameters. Figure 15 presents the part of the sequence diagram associated with the core of the analysis process. The main task is launched by the “processTrace” event. Then the key subtask is to compute the occurrence vector for each class that is represented by the “getOccurrenceVectorMap”.

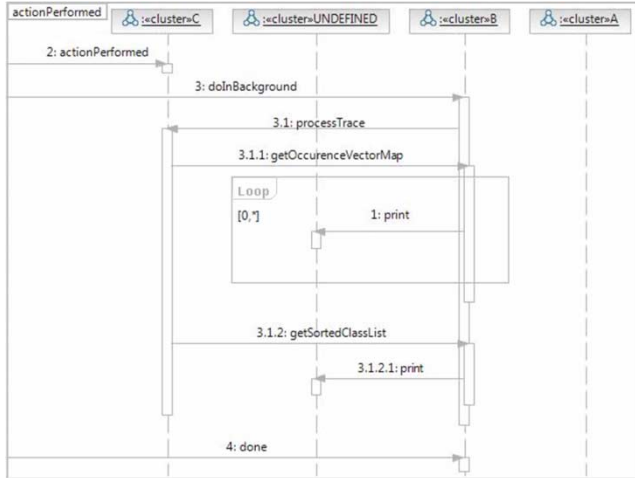


Figure 15: Sequence diagram for the analysis step

Finally the event “getSortedClassList” is used to display the results on the screen. In this example, our “abstraction” technique was able to retain and display the key events in the system. As a second test we applied the technique to an industrial system with 600 classes and whose execution trace had more than 600’000 events. The trace involved 169 classes and the clustering algorithm found 35 clusters. The compression technique ended up with about 200 events. This represents a compression factor of 3000. Although we could still display the resulting sequence diagram in RSA, we cannot show it in this paper since this would take about 8 pages. Again, the resulting event set showed the key working steps of the system.

7. CONCLUSION

In this paper we have showed that our software clustering technique, which is based on the dynamic analysis of the methods called when executing a scenario, allow us to automatically build an “abstract” view of the working of a system. In particular we are able to show that the main steps of a complex processing have been automatically revealed in a sequence diagram. Moreover, the generated sequence diagram seems to be close to the one we would actually draw by hand to explain the working of the system. Although this important feature needs be confirmed by more experiments it is very encouraging. This result has been obtained because of our very dynamic analysis technique coupled with the automatic generation of sequence diagrams. Here again are the main steps of our approach:

- An execution trace is generated when executing some scenario of business value (use-case). We only keep the events between application classes.
- The clusters that are identified are strongly linked to business functions. In a sense they represent chunks of elementary functionality;

- The working of the system is abstracted by displaying only the method called between the clusters.
- The resulting event set is further compressed using event patterns repetition compression.

Our cluster identification technique is to be contrasted with the ones that try to identify the components based on some software engineering metrics such as high cohesion and low coupling. As explained in paragraph 2, we are not targeting the recovery of any structural view of the software. Rather, we wish to generate a view that would support the understanding of software behavior. Therefore the conceptual grouping of classes as well as the analysis technique cannot rest on static analysis techniques. With the latter, there is no hope that the component will represent steps of some complex behavior related to the business function. The comparison of the dynamic clusters with the static structure of the program (for example its packages) is useful to show where the responsibilities pertaining to some behavior are actually implemented. But, as we already explained in paragraph 2, the criteria that lead to the static structure of the code are usually not based on the working of scenarios. Therefore, there little hope to end up with a set of functional components that would match the packages of the source code.

Our technique has been integrated in the IBM’s RSA environment as an Eclipse plugin. In summary, the key contribution of this paper is to show that the automatic generation of an abstract view of a legacy system is indeed possible and we presented the way we achieved this. To refer back to the definition of program understanding by Biggerstaff at al. [8] we generated a view that would let an engineer understand the program behavior as interacting abstract functional components that are qualitatively different from the tokens of the source code i.e. the classes. Our approach therefore represents a step toward the true support of program understanding at higher level of abstraction.

So far, we only worked on single scenarios at the time (i.e. the main flow of a use-case). As future work we will develop multi-scenarios clustering and analysis techniques with alternatives. This may allow us to identify reusable functional components. Moreover we will integrate some zooming feature to be able to perform the same clustering analysis on the functional components themselves. Then we will develop a new view with the links between the different abstraction levels.

8. RELATED WORK

In the literature, many techniques have been proposed to recover the structure of a system by splitting it into components. They range from document indexing techniques [22], slicing [34] to the more recent “concept analysis” technique [27][30] or even mixed techniques [17]. All these techniques are static i.e. they try to partition the set of source code statements and program elements into subsets that will hopefully help to rebuild the architecture of the system. But the key problem is to choose the relevant set of criteria (or similarity metrics) [35] with which the “natural” boundaries of components can be found. In the reverse-engineering literature, the similarity metrics range from the interconnection strength of Rigi [24] to the sophisticated information-theory based measurement of Andritsos and Tzerpos [1][2], the information retrieval technique such as Latent Semantic Indexing [22] or the kind of variables accessed in formal concept analysis [31]. Then, based on such a similarity metric, an algorithm decides what element should be part of the same cluster [21]. In their work, Xiao and Tzerpos compared several clustering algorithms based on dynamic dependencies. In

particular they focused on the clustering based on the global frequency of calls between classes [37]. This approach does not discriminate between situations where the calls happen in different locations in the trace. This is to be contrasted with our approach that analyzes where the calls happen in the trace. Very few authors have worked on sampling or segmentation techniques for trace analysis. One pioneering work is the one of Chan et al. [9] to visualize long sequence of low-level Java execution traces in the AVID system (including memory event and call stack events). But their approach is quite different from ours. It selectively picks information from the source (the call stack for example) to limit the quantity of information to process. The problem to process very large execution traces is now beginning to be dealt with in the literature. For example, Zaidman and Demeyer proposed to manage the volume of the trace by searching for common global frequency patterns [38]. In fact, they analyzed consecutive samples of the trace to identify recurring patterns of events having the same global frequencies. In other words they search locally for events with similar global frequency. This is then quite different from our approach that analyzes class distribution throughout the trace. Another technique is to restrict the set of classes to “trace” like in the work of Meyer and Wendehals [23]. In fact, their trace generator takes as input a list of classes, interfaces and methods that have to be monitored during the execution of the program under analysis. Similarly, the tool developed by Vasconcelos, Cepêda and Werner [33] allows the selection of the packages and classes to be monitored for trace collection. In this work, the trace is sliced by use-case scenarios and message depth level and it is then possible to study the trace per slice and depth level. Sartipi and Safyallah [26] use a patterns search and discovery tool to separate, in the trace, the patterns that correspond to common features from the ones that correspond to specific features.

8. REFERENCES

- [1] Andritsos P., Tzerpos V. 2003. Software Clustering based on Information Loss Minimization. Proc. IEEE Working Conference on Reverse engineering. 2003
- [2] Andritsos P., Tzerpos V. 2005. Information Theoretic Software Clustering. IEEE Trans. on Software Engineering 31(2). 2005
- [3] Anquetil N., Lethbridge T.C. – Experiments with Clustering as a Software Remodularization Method. Proc IEEE WCRE, 1999.
- [4] Bass L., Clements P., Kazman R. 2003. Software Architecture in Practice, 2nd edition. Addison-Wesley Inc.
- [5] Belady L., Lehman M. 1976. A Model of Large Program Development. IBM Syst. Journal 15(3), pp. 225-252.
- [6] Bennett C., Myers D., Storey M.-A., German D. 2007. Working with ‘Monster’ Traces: Building a Scalable, Usable Sequence Viewer. Proc. of the 3rd International Workshop on Program Comprehension through Dynamic Analysis.
- [7] Bergey J., Smith D., Weiderman N., Woods S. 1999. Options Analysis for Reengineering (OAR): Issues and Conceptual Approach. Software Engineering Institute, Tech. Note CMU/SEI-99-TN-014, 1999.
- [8] Biggerstaff T. J., Mitbander B.G., Webster D.E. 1994. Program Understanding and the Concept Assignment Problem. Communications of the ACM, CACM 37(5), 1994.
- [9] Chan A., Holmes R., Murphy G.C., Ying A.T.T. 2003. Scaling an Object-oriented System Execution Visualizer through Sampling. Proc. of the 11th IEEE International Workshop on Program Comprehension (ICPC’03).
- [10] Clements P. et al. 2002. Documenting Software Architectures: Views and Beyond, Addison Wesley.
- [11] Dugerdil Ph. 2007. Using trace sampling techniques to identify dynamic clusters of classes. Proc. of the IBM CAS Software and Systems Engineering Symposium (CASCON), 2007
- [12] Dugerdil Ph., Jossi S. 2008. Empirical Assessment of Execution Trace Segmentation in Reverse-Engineering. Proc. ICISOFT 2008, Porto Portugal.
- [13] Dugerdil Ph., Jossi S. 2009. Computing Dynamic Clusters. 2nd Indian / ACM Conference on Software Engineering (ISEC) 2009, Pune, India, February 23-26.
- [14] Hamou-Lhadj A., Lethbridge T. 2004. A Metamodel for Dynamic Information Generated From Object Oriented Systems. Electronic Notes in Theoretical Computer Science 94, pp.59-69.
- [15] Hamou-Lhadj A., Lethbridge T. 2006. Summarizing the Content of Large Traces to Facilitate the Understanding of the Behavior of a Software System. Proc. of the IEEE Int. Conference on Program Comprehension (ICPC’06).
- [16] Hamou-Lhadj A., Lethbridge T.C 2002. Compression Techniques to Simplify the Analysis of Large Execution Traces. Proc. of the IEEE Workshop on Program Comprehension (IWPC).
- [17] Harrman M., Gold N., Hierons R., Binkeley D. 2002. Code Extraction Algorithms which Unify Slicing and Concept Assignment. Proc IEEE Working Conference on Reverse Engineering (WCRE’02).
- [18] www.hibernate.org
- [19] www.springsource.org
- [20] Koskinen J. – Software Maintenance Costs. University of Jyväskylä, Finland, <http://users.jyu.fi/~koskinen/smcosts.htm>
- [21] Maqbool O., Babri H.A. 2007. Hierarchical Clustering for Software Architecture Recovery. IEEE Trans. On Software Engineering, Vol. 33, No. 11.
- [22] Marcus A. 2004. Semantic Driven Program Analysis. Proc IEEE Int. Conference on Software Maintenance (ICSM’04).
- [23] Meyer M., Wendehals L. 2005. Selective Tracing for Dynamic Analyses. Proceedings of the 1st International Workshop on Program Comprehension through Dynamic Analysis (PCODA’05).
- [24] Müller H.A., Orgun M.A., Tilley S., Uhl J.S. 1993. A Reverse Engineering Approach To Subsystem Structure Identification. Software Maintenance: Research and Practice 5(4), John Wiley & Sons. 1993
- [25] Repond J. 2009. Génération de diagrammes UML à partir d’une analyse dynamique. Bachelor Thesis, HEG, Univ of Applied Sciences of Western Switzerland, Geneva.
- [26] Sartipi K., Safyallah H. 2006. An Environment for Pattern based Dynamic Analysis of Software Systems. Proceedings of the 2nd International Workshop on Program Comprehension through Dynamic Analysis (PCODA’06).
- [27] Siff M., Reps T. 1999. Identifying Modules via Concept Analysis. IEEE Trans. On Software Engineering 25(6). 1999.

- [28] Simon H.A. 1969. The architecture of complexity. In: The Sciences of the Artificial, MIT Press. (Reprinted in 1981)
- [29] Tilley S.R., Santanu P., Smith D.B. 1996. Toward a Framework for Program Understanding. Proc. IEEE Int. Workshop on Program Comprehension.
- [30] Tonella P. 2001. Concept Analysis for Module Restructuring. IEEE Trans. On Software Engineering, 27(4).
- [31] Tonella P. 2003. Using a Concept Lattice of Decomposition Slices for Program Understanding and Impact Analysis. IEEE Trans. On Software Engineering, 29(6).
- [32] Tzerpos V., Holt R.C. 2000. ACDC : An Algorithm for Comprehension-Driven Clustering. Proc. IEEE Working conference on reverse Engineering (WCRE).
- [33] Vasconcelos A., Cepêda R., Werner C. 2005. An Approach to Program Comprehension through Reverse Engineering of Complementary Software Views. Proceedings of the 1st International Workshop on Program Comprehension through Dynamic Analysis (PCODA'05).
- [34] Verbaere M. 2003. Program Slicing for Refactoring. MS Thesis, Oxford University.
- [35] Wiggerts T.A. 1997. Using Clustering Algorithms in Legacy Systems Remodularization. Proc IEEE Working Conference on Reverse Engineering (WCRE '97).
- [36] Wirfs-Brock R., McKean A. 2003. Object Design, Roles, Responsibilities and Collaborations. Addison-Wesley.
- [37] Xiao C., Tzerpos, V. 2005. Software Clustering basd on Dynamic Dependencies. Proc. of the IEEE European Conference on Software Maintenance and Reengineering (CSMR'2005).
- [38] Zaidman A., Demeyer S. 2004. Managing trace data volume through a heuristical clustering process based on event execution frequency. Proc. of the IEEE European Conference on Software Maintenance and Reengineering (CSMR'2004).