AUTOMATIC ALERT SYSTEM: IMPROVING INFORMATION MANAGEMENT ON CONSTRUCTION PROJECTS

Tyler Bushnell, busht@stanford.edu
Center for Design Research, Mechanical Engineering Department, Stanford University, United States

Aman Neelappa, aman313@stanford.edu
Computer Science Department, Stanford University, United States

Reid Senescu, rsenescu@stanford.edu
Center for Design Research, Mechanical Engineering Department, Stanford University, United States

Martin Fischer, fischer@stanford.edu
Center for Integrated Facility Engineering, Department of Civil and Environmental Engineering, Stanford University, United States

Martin Steinert, martin.steinert@ntnu.no
Department of Engineering Design and Materials, Faculty of Engineering Science and Technology, Norwegian University of Science and Technology

ABSTRACT

A construction project has too much information for each team member to be aware of other members' interactions with the information. Most information is contained in files. Despite file syncing, version control, and check-in/check-out technologies, complex information flows result in incorrect or inconsistent information delivery to the construction site. Our paper proposes the Automatic Alert System methodology for using a log of team members' interactions with an information management tool to notify users of relevant changes. The methodology builds on previous research that automatically discovers the dependencies between files based on team members' interactions with the files. Our paper proposes a system of relevant alerts to team members when a dependent file changes. This system is simulated using a case study from an actual construction project. Results show that alerts may increase awareness of file changes without overwhelming users. Stakeholders qualitatively verified the usefulness of the simulated findings.

Keywords: communication, collaboration, information management, process modeling

1. INTRODUCTION

The information is out there, but not always accessible, and usually there is no way to find it. Information is also changing all the time as the project develops, resulting in outdated files and records that must be updated. Usually files can be examined to see that they've been updated, or that new versions have come up, but there is more to the story than an updated file. Researching automatically generated file dependency is a new topic of research, and file dependencies can be gathered using a network file tool to automatically generate file dependency information (Senescu et. al. 2012). File dependency data can then be linked to professional users on a construction project, and these users can be made aware of more than just the known file changing, but also any other files that may lead to changes in the important file. These alerts can be used to help companies and professionals track changes of high priority files, or manage information to avoid problems on-site and rework.

At the time of writing the authors are only aware of the work by Senescu et. al. for the creation of file dependency data. This paper is the first paper to propose using this dependency data to solve problems on a construction site.

Section 2 will describe an actual case from a hospital construction project in California since 2007. Section 3 proposes a system for automatically generating alerts for users. Section 4 describes the validation case and results using actual archived project data.

2. EXISTING STRUGGLES WITH MANAGING INFORMATION

This section describes challenges faced by a project team designing and constructing a California hospital. We gathered information from interviews and logs of team member interactions with the information management tool (Projectwise). Section 2.1 describes the information flows the team agreed to follow in the context of Design Process Management literature. Sections 2.2 and 2.3 describe the results of our case study which examined two challenges the team faced when implementing the intended information flow. This paper proposes the Automatic Alert System (AAS) methodology to address these two challenges.

2.1 Existing information flows the team intended to follow

The project team designed and modeled a new hospital in three years before breaking ground in 2011. During the project (and prior to our observation), the companies on the team agreed to follow the information flows described in this section.

Each of the companies involved in the pre-construction phase created their portions of the Building Information Model (BIM) and checked for conflicts with other companies each day in the collocated "big room." Once the team completed coordination for a portion of the building, the team agreed to freeze design and the portion was then saved to the master model file. After the team decided to freeze a portion of the model, companies that desired a change to the portion were required to resolve any clashes that the change caused in the master model. Companies continuously iterated on portions of the model, updating the master model with the most current design. The team archived the master model frequently as there was no explicit communication to the entire team when a particular company updated the master model.

If a company updated the master model with a change that was later discovered to cause problems, the project manager or project coordinator searched for the change in the archived master model files until the change was located. That is, sometimes the project manager or coordinator knew that there was a problem with the current design, but did not know which change resulted in the problem. This searching for the cause of the problem was a manual time consuming process, but it was necessary, so the problem could be resolved before construction.

The design rationale literature has addressed this challenge with numerous methodologies for documenting design decisions. It would also have been easy to create an information flow with a stop gate requiring changes to be approved before the companies updated the master model file. However, in practice it was simply too burdensome to explain each update when a file was updated multiple times per day by each company. The team likely did not create a process for documenting changes to the master model file for this reason. Also, an approval process would inhibit the fast iterative collaboration promoted by the "big room" and other efforts to strive toward integrated concurrent engineering. Consequently, the burden of communicating information flows lied with the team members. According to the project manager, "If you move things, you need to let everyone know there's a new file out there. If you move something and create a clash, clear your own clashes. If you need help, reach out." Also, "Everything [the subcontractors] do matters. How their designs interact matters. Even if [coordination] may seem easier at the time, when they don't tell everyone [about their updates] it becomes an issue." Of course, each company has continuous deadlines, and communicating information flows is not a priority. Again, literature both emphasizes the importance of communicating information flows (see summary of this research in Senescu et. al. (2013)) and simultaneously admits that despite this importance, teams rarely communicate information flows in practice (Conklin and Yakemovic 1991; Ishino and Jin 2002; Moran and Carroll 1996).

In addition to problems caused by the lack of communicating information flows, the unavailability of information and latency in delivery of information also caused problems. Each subcontractor (sub) had its own

private folders in the information management tool that were used for all official project documentation and for sharing files. The subs controlled these private folders, and they usually password protected the folders from the other companies on site, including the general contractor. When the files were complete, the sub shared the private files with the entire team in a public folder and updated the master model files that the entire team used as reference. Thus, there was necessarily a delay between a sub updating their own files, and other stakeholders affected by the changes knowing about the change. The next two sections describe two struggles that occurred as subs attempted to follow the above information flows.

2.2 Subcontractors struggle with outdated information

In one implementation of the intended information flow, the plumbing sub coordinated his private file with the design team's model in Navisworks (a building model coordination and viewing application). The project team approved the coordinated model as ready for construction. During the construction phase, the plumbing sub modified his private model, because the master public model was frozen three months prior to the start of construction phase. The plumbing sub and other subs began construction, but the plumber did not publish his changes to the master model. In fact, the plumbing sub had multiple working versions of his private models. Meanwhile, the other subs were coordinating their work with a master model file that was out-dated, because it did not have any of the latest plumbing updates.

According to the intended information flow described in Section 2.1, it was the plumbing sub's responsibility to keep the master model files up-to-date. The plumbing sub had created the correct information, and others would have been able to coordinate if they knew that updated information existed or even if they were able to look for places where updated information could potentially exist. The plumber was not intentionally hiding the information – it was merely a missed step in an extremely complex information flow. Also, it was in his best interest to alert others of the changes so that they could respond, and problems could be avoided to save money and time.

2.3 Struggles managing subcontractors who try (or don't try) to follow information flows

The process in Section 2.1 may not be the most efficient information flow, but it generally worked as expected for most portions of the hospital and most of the subs. However, the flow lacked both redundancy and robustness to avoid even bigger inefficiencies when subs failed to follow the process. Around May 2012 the HVAC/Mechanical sub had to change the electrical access panels on the 5th and 6th floor portions of the model. In iterating to find a coordination solution on their private files, the HVAC/Mechanical sub hid and sometimes removed access zone objects. They did this with good intentions since it enabled them to avoid tedious warnings about clashes while attempting to resolve clashes. However, they never fully coordinated their change on their private file. The sub then updated the master model with the change. This update was uncoordinated (i.e., the updated master model had clashes). Normally, the General Contractor (GC) would detect this clash in the master model and then follow the burdensome process described in Section 2.1 to find the cause. This flow was not efficient, but at least they could resolve the clash before construction. In this case, no clash was detected, because the access zones were not in the model, so the clash went undetected until construction. In other words, the team was constructing parts of the building without zones – requiring manual rework. "There were supposed to be 'no fly zones' and now, during construction, access zones were turned off, and we've had build issues and had to move stuff in the field… [accessibility] was still no good." During construction, the GC discovered the clash, but did not know the root cause. Then, the GC realized the problem occurred repeatedly – not just one change to the master model, but many changes.

Eager to identify the root cause (still unknown to the GC), the BIM coordinator and Project Manager began checking the BIM archives and looking for when and where the problem started. Our data analysis shows this tedious search for the root problem in Figure 1. Late 2012 the Project Manager discovered that the hidden or deleted access zones were the root of the problem. He informed the sub. However, the PM reported that a particular sub continuously updated uncoordinated models without access zone objects despite requests from the GC. Aware of this human breakdown in the agreed upon process, the Project Manager continuously searched for

instances when this particular sub updated the master model. Figure 1 shows this persistent and tedious searching for changes.

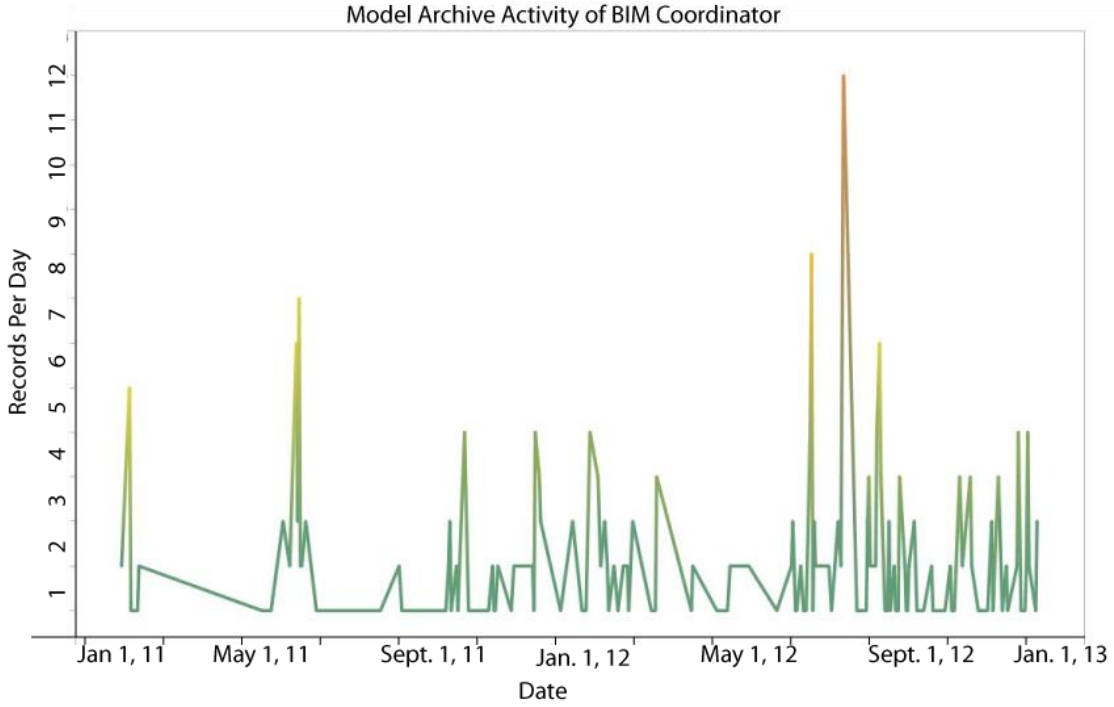

Figure 1: The file logs show that there is an increase in checking the published files between July and September 2012, and then a high frequency of activity afterwards.

The project manager proposed three potential solutions to avoiding this rework. First, one could take the approach of changing human behavior: "Subcontractors are responsible for cleaning up clashes and keeping their files. How do you stop individuals that just do something?" Second, "there could have been better control" of the information flow by requiring approval before enabling subs to update the master model. Third, a solution could simply focus on intelligently revealing changes: "If I knew the access zone had been touch or moved, that would have been great." As researchers, we believe changing human behavior to follow information flow perfectly is nearly impossible. Adding additional control would be costly and would strip the team of the many benefits that come with collaborating in a collocated environment with a master BIM. Thus, we take the third approach. Humans (even some of the most highly respected subs in the world) make mistakes when following information flows. Our goal is to bring transparency to these flows, so that the team can adapt to changes and mistakes with agility to avoid costly rework.

3. METHODOLOGY FOR ALERTING USERS

3.1 Generating dependencies between files

The project used a shared information management system called Bentley ProjectWise that logs every action a user performs on a file. All companies used this file system to store and share official files. This file system creates a log file for each file that includes information about who is using the system, what they are doing, and when. Using this information we can determine who is reading which files and who is editing which files. The file logs used in this study represent system usage from January 2009 to December 2012 for altogether 2109 users from 24 different projects, including the project discussed in the above case study. For the project in this study there are 1,355,257 log entries that describe activity of 141,952 files. Each log entry includes exact project, path, timestamp, username, user description, filename, action ID, and version. The username includes a prefix with an organizational ID that indicates the company that the user is affiliated with. The action ID tells what the user did,

including “viewed,” “created,” “Updated,” “checked-in, changed,” “checked-in, not changed,” “deleted,” and “uploaded.” These action IDs are sorted into “Write” and “Read” actions to easily distill how users are interacting with the files.

The log information can be used to generate dependencies between files in real time, and create dependency data for the network (Senescu 2012). The network looks at how people are interacting with files, and the time proximity of use. This will be accepted as the method for creating links between files that includes dependency. For example, if UserA views File1 before he edits File2, and this happens repeatedly, it would be said that File2 depends on File1. That is, what happens in File1 effects File2 and it is important to know if File1 changes because a change could lead to a change in File2.

3.2 Establishing alerts

Users need to know when files change that effect their work. This requires an Automated Alert System (AAS) to create an alert log that tracks when a file may be out of date. The dependency data is already based on the users that generate it, so those users are then associated with files when they read or write them. For every user action on the file system, the AAS checks and updates the file dependency, and then creates an alarm in the system for every user that is connected with the dependent files. The figure below is a flow diagram for how the AAS generates user links and alerts.

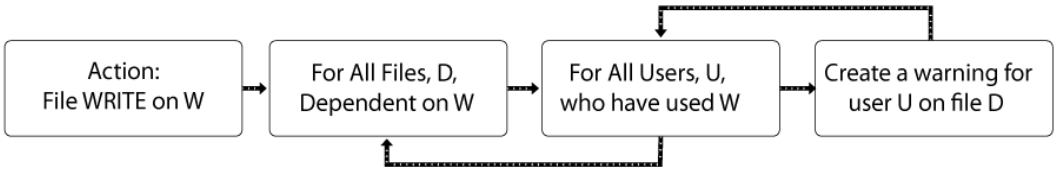

Figure 2: Explanation of write actions creating warnings for users

It is important to note here that no users have received any alerts at this point. The alerts live on the system and are not automatically broadcast and are instead are created as a starting point to see all of the dependencies of people and files in the system.

3.3 Creating an automated alert system

The alarming system is based on the system alerts. If UserA reads File1 that has an alert on it for them, that is, another File2 has changed that may have impacts on File1, then UserA will receive an alarm notifying him that File1 may be out of date. Figure 3 provides a visual example of this sequence.

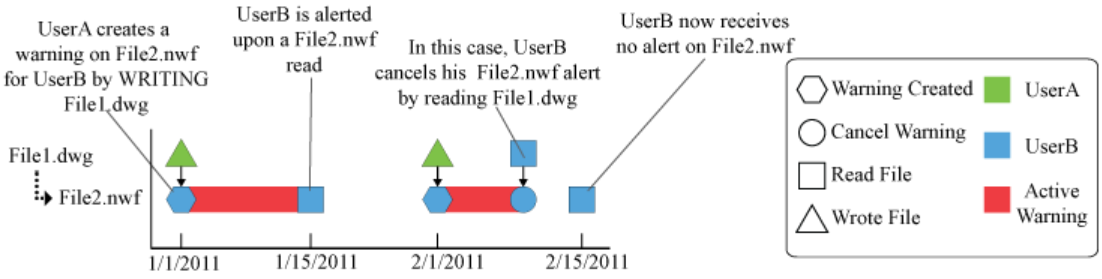

Figure 3: This timeline describes two examples of UserA creating an alert for UserB by editing a file that is associated with UserB.

The alerts will contain information relevant to UserA including the filepath for File2, UserB information whom edited File2, and a timestamp for when this was done. Enough information that UserA can make a decision on how to proceed knowing who changed what file, and when.

The alerts can then be filtered down more by noticing that if UserA has already viewed File2 since it's been updated then UserA should not receive an alert when he opens File1, because he is assumed to already know it has changed. For the purposes of this paper this will be called an "unalert." An example of how an unalert works is included in Figure 3.

Factoring in unalerts, each user only receives an alarm notification if they open a file with an alert on it without generating the unalert beforehand. The alarm can be given as something built into the file system software, or it could use existing methods like email.

4. RESULTS AND VALIDATION

The results in this section are based on actual archived file logs with the companies involved in the case study. Alerts and warnings generated are based on the data from the project, and therefore accurately simulate the results of that project.

4.1 Assisting subcontractors with outdated information

In the first case presented in Section 2.2, the plumber was editing his own AutoCAD drawing files which a Navisworks file was dependent on. Because he failed to update the master Navisworks file which was used for coordination, the changes he was making were unaccounted for by other subcontractors, and eventually clashes began happening in the field.

Now, assume the team had AAS. The set of coordination files would have dependency links with the plumber's drawing files because of the associations that had been produced during the design phase. When the plumber updated his files, an alert was created for users of dependent files, including the coordination files. The plumber was using multiple files, but the dependencies on all of those files lead to connections with some coordination files.

None of the other subcontractors could access the plumber's drawings because they were protected in his own folders that he manages, so no unalerts were created for dependent files.

As trades updated dependent files, the coordination files gathered many warnings. The project progresses in the following months, and as building began, the project manager is aware that the plumber has changed his files because of the alerts he received from the plumber.

Figure 4 shows the superintendent would have been alerted by the plumber's changes to files linked to DesignPhase.nwf at the end of December 2010 when the model was frozen. The GC and BIM coordinator already knew that the plumber was making changes, so they could have ensured they received all of those changes. There was a series of alerts related to L03-Area 01.nwf, one in March, and three in July. This file pertains to the area the BIM Coordinator identified as "getting messy," and Figure 4 shows that he could have been alerted about the additional files that were linked months in advance.

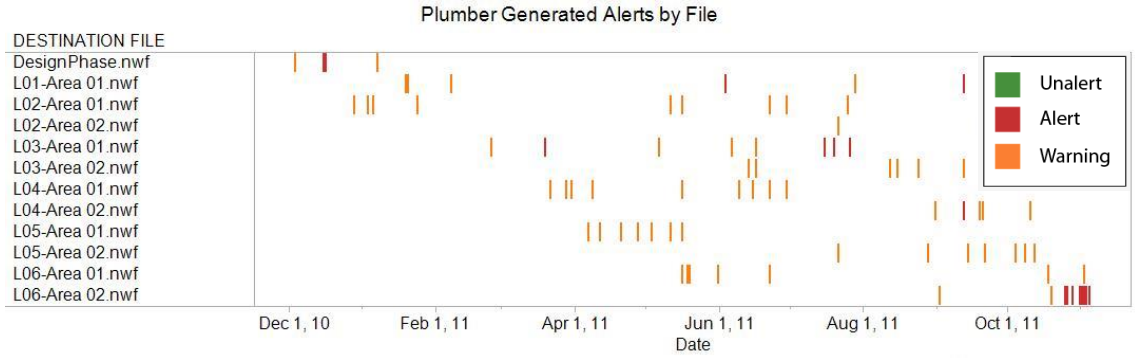

Figure 4: Files that had alerts for the Project Manager and BIM Coordinator caused by the Plumber's writes

In this case the BIM Coordinator discovered that the plumber was working from multiple versions before there were any clashes in the field, and the plumber can correct his process before it costs schedule or budget.

4.2 Assisting management with subcontractors and human error

Several subs updated the master Navisworks coordination files discussed in Section 2.3. When clashes exist on the coordination file it is tedious to figure out who made the changes. This makes it difficult to identify root causes and respond correctly. Because the project manager was unable to track who was changing the model in an efficient way, he was forced to adopt tedious methods for correction that responded to rework, rather than preventing it.

AAS could have created alerts as the subs edited their files. When the project coordinator opened the master coordination file, he would have received an alert that the subs' private files have changed, and he would know right away who changed them. When the model had a clash in it, he could have advised the HVAC/Mechanical subcontractor to correct the clashes. This can be seen in Figure 5 as the first alerts set off in June by the Fire Protection Sub and the HVAC/Mechanical Sub.

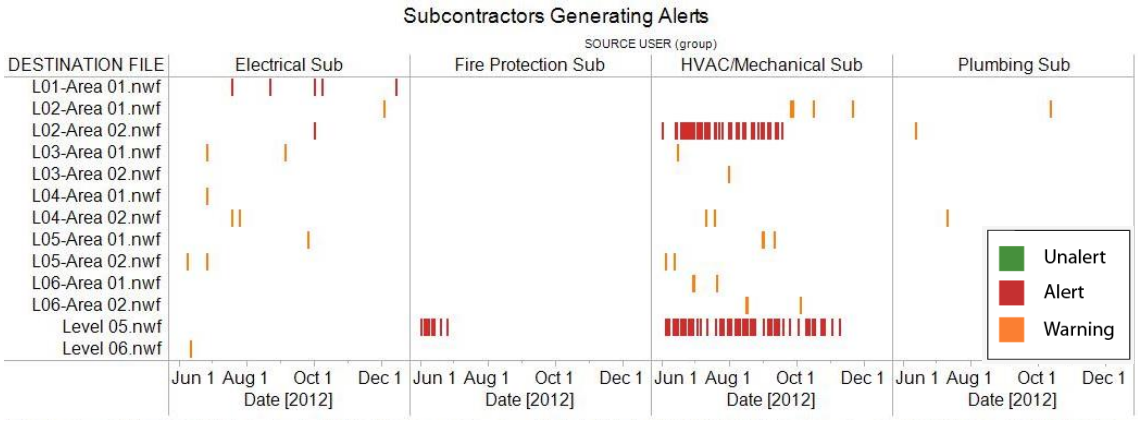

Figure 5: The AAS timeline for each sub involved in this case shows a disproportionately large number of alerts generated by the HVAC/Mechanical sub.

When the same subcontractor continued to cause clashes in the model after updating it with their changes, the project coordinator could begin to pay special attention to alerts regarding the subcontractor. The coordinator could follow up every time he received an alarm to make sure all access objects in the model were correct and up to date. While this is inconvenient for him, it prevents him from needing to constantly check who updated the model. This adaptation to the process on this team leads to more efficient behaviors and prevented rework on-site.

4.3 Project-wide results

The macro-level results of an implemented AAS considers all users on the project. Figure 6 shows how many files generate alerts, unalerts, and warnings each day of the project. The dip in the center of the graph corresponds with the transition between design phase and build phase, and a period which the model was frozen from changes. The graph aligns with what is expected. Considering the people that are involved on the project, Figure 7 shows the average alerts per user over the length of the project. This figure only takes into account users who received an alert that day. There are times in the design phase that average over ten alerts per day, but when the build phase starts the average drops, and stays around five alerts per day for a user.

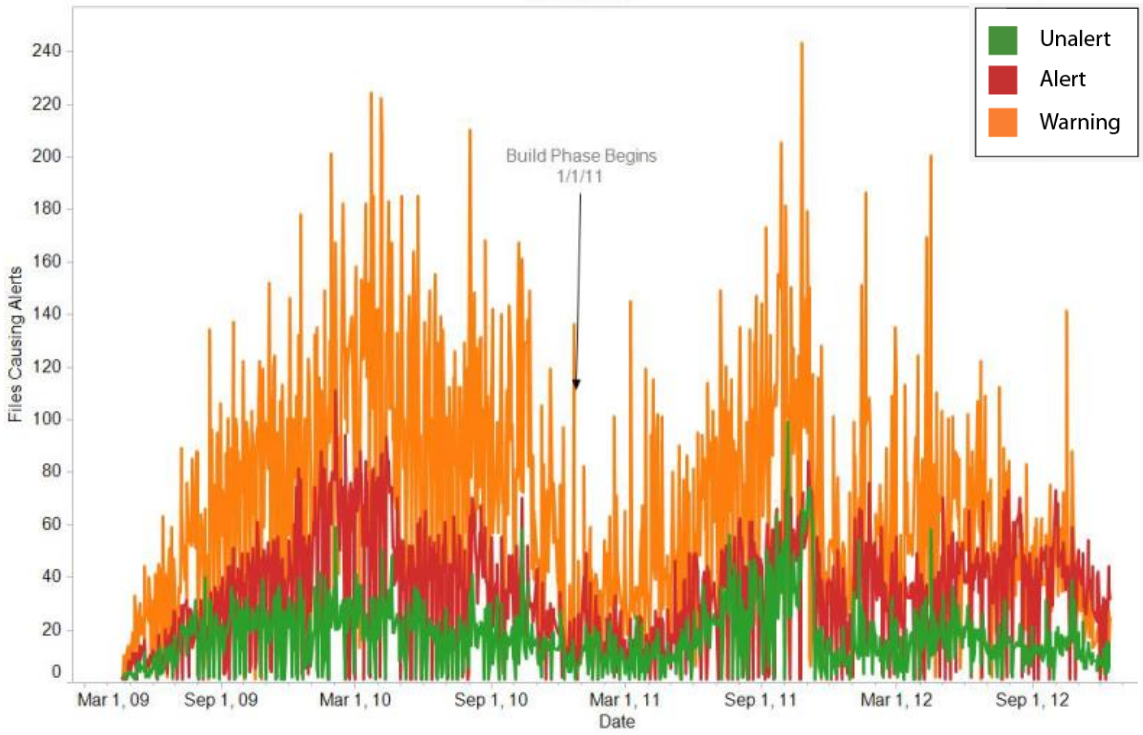

Figure 6: The total number of distinct files that are generating AAS activity each day. The maximum is equivalent to 0.0017% of the project files.

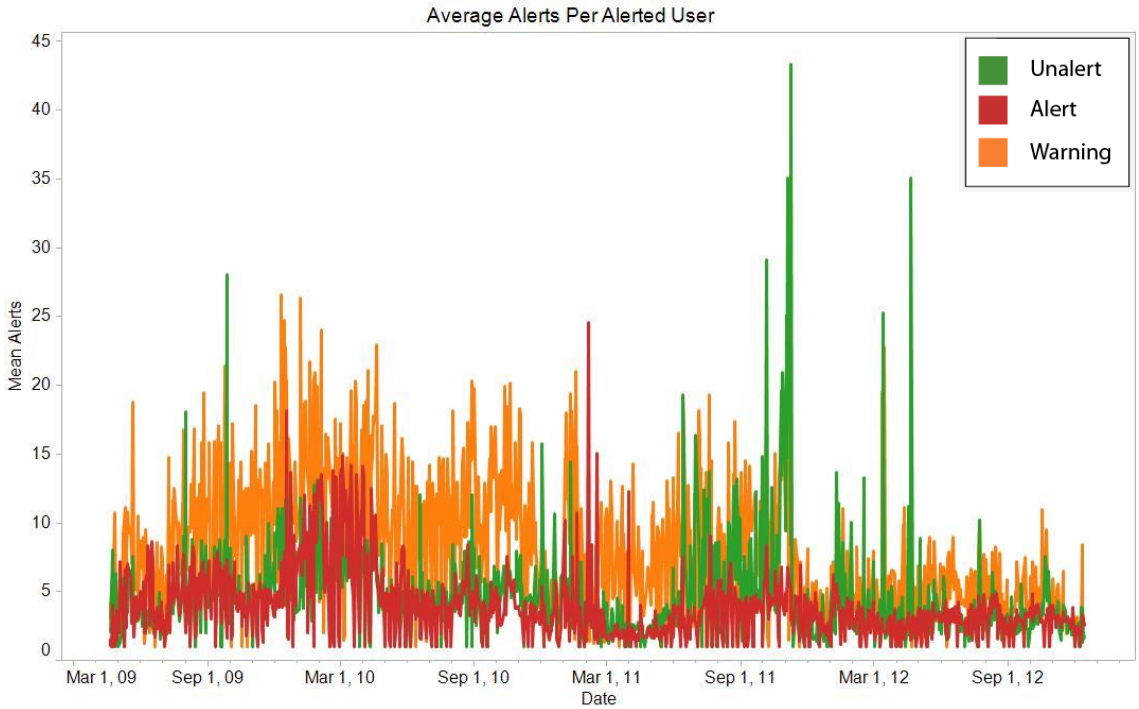

Figure 7: The mean number of AAS alerts for users who received an alert, per day.

To understand more about what alerts look like for each user, Figure 8 includes three user profiles that are discussed in this paper.

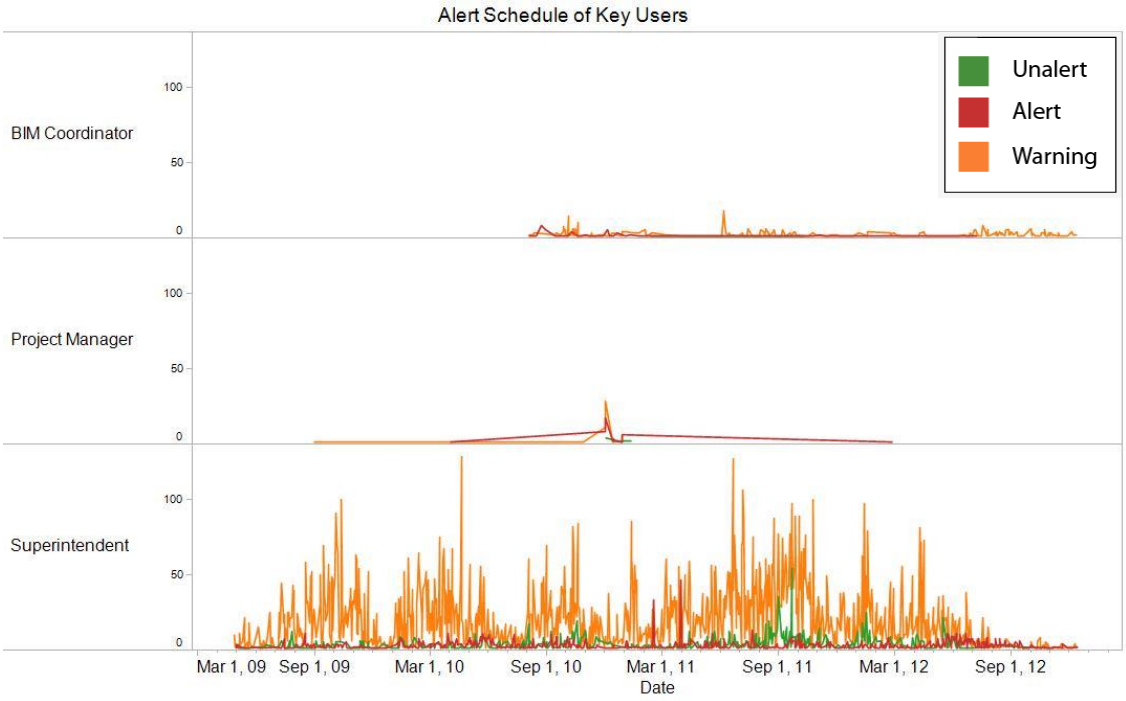

Figure 8: The simulated alert schedule of three important team members

5. CONCLUSION

We described two struggles that a case study project team encountered when managing information on a hospital project. The struggles described in the case study are only two examples that interviews revealed are representative of many information management and collaboration struggles both on the case study project and on other projects. Thus, addressing these two examples would actually result in large gains in efficiency for the case study project and the overall construction industry. To address these struggles, we proposed the Automatic Alert System (AAS) methodology. AAS uses team members' interactions with files to notify users of relevant changes.

In applying AAS, users would be able to have greater control over who and how they receive alarms. These types of incremental improvements on the system demonstrate how collaborative construction projects will be organized. Given that collocation and big-room methodologies leads to so many improvements, these are the issues that will need to be addressed when other, larger blocks to information transparency are already removed.

The evidence present in this paper suggests that the file logs have sufficient information for us to be able to infer and dependencies in files which can then be used to create a system where a warning can be generated to prevent miscommunication and possible rework. Further, we have also shown that the system thus set-up only sends a tractable number of warnings to a user on average. There are a few areas that we would like to improve in future. One is that AIDA can be improved by collecting user data more carefully. On the other hand, we will be collecting more data to enable us to apply machine learning techniques for finding file dependencies. We would also like to study more cases similar to those we studied in this paper and better characterize the kinds of scenarios such a system would be of help.

This paper only presents the AAS system as a theoretical possibility, and it has not been fully implemented for validation. Additionally it is based on the success of the file dependency network proposed in AIDA, or a system for linking dependencies between files on the shared system. Even this depends on the way a project interacts with the shared file system for documenting all documentation.

REFERENCES

Conklin, J., & Yakemovic, K. (1991). "A process-oriented approach to design rationale". *Human Computer Interactions*, 6, 357-391.

Ishino, Y., & Jin, Y. (2002). "Estimate design intent: a multiple genetic programming and multivariate analysis based approach". *Advanced Engineering Informatics*, 16(2), 107-125.

Moran, T. P., & Carroll, J. M. (1996). "Overview of design rationale". *Design Rationale: Concepts, Techniques, and Use*, Lawrence Erlbaum Assoc., Mahwah, NJ, 1-19.

Senescu, R., Head, A., Steinert, M., & Fischer, M. (2012). "Generating a network of information dependencies automatically". *14th International Dependency and Structure Modeling Conference Proceedings*, Carl Hanser Verlag, Munich, Germany, 139-144.

Senescu, R., Haymaker, J., Meza, S., & Fischer, M. (2013). "Design process communication methodology: improving the effectiveness and efficiency of collaboration, sharing, and understanding". *Journal of Architectural Engineering*, in press.