Advanced Software Design
The MATCH System Specification

Prepared by Dave Clarke
Update 2016-2019: Kiko Fernandez-Reyes

October 10, 2019

Abstract

This document specifies the requirements of the MATCH system, which will be used as the basis of the design activities in Advanced Software Design. Firstly, a vision of the MATCH software system is given along with a number of scenarios in which MATCH could be used. This is followed by a specification of the core MATCH infrastructure. Then, a description of various configurations of the MATCH system are given. An ideal MATCH infrastructure should be flexible enough to support all configurations. Finally, a description of what your group needs to do is given.

1 Vision — MATCH: Connecting People

MATCH is an software system for connecting people. It matches a human *requester* (someone in need of a service or information) with one or more *responder* (people who can provide that service or information). MATCH uses domain-specific intelligent matching functionality *and* learning to find effective connections between requesters and responders in a community. MATCH is highly configurable and can easily be extended to support different kinds of service. MATCH derives domain-specific knowledge about the connections between people in these distributed heterogenous settings, to better facilitate the connection of people.

Scenario: Senior Care

Jock, a senior citizen, is no longer as mobile as he used to be. One morning he knocks a pot plant[1] off the window sill and soil spills all over the floor. The cleaner isn't due for another 6 days, which means that the soil will be on the floor until then and the plant will likely die.

Fortunately, Jock has signed up with MATCHCare, a service based on the MATCH system for connecting seniors with local people able to help, for a

[1] For medicinal purposes.

short time, on short notice. Jock opens the MATCHCare Requester App on his iPhone and starts a new request. The interface is simple. He indicates that the job will take between 5 and 10 minutes and that it is non-urgent, but should be done today.

His request is sent to the MATCHCare Server, which searches through its database of possible responders. A number of candidates are selected: Robin, who lives on the same street as Jock and has helped Jock before; Selma, a student who is often in the neighbourhood; and Gunde, an unemployed former Olympic skier.

The MATCHCare Server sends a message to Robin via his MATCHCare Responder App. Robin opens the message immediately, but as he is visiting his uncle in Östersund, he is unable to help, and declines.

The MATCHCare Server then sends a message to Selma via iMessage. After 30 minutes, Selma fails to responds so MATCHCare Server sends a message cancelling the request.

The MATCHCare Server then contacts Gunde via his Apple Watch. Gunde has just finished a run, so he'll be able to help after having a shower. He accepts the request via the Apple Watch MATCHCare Responder App, indicating that he'll be there in 30 minutes.

The MATCHCare Server then sends a message to Jock, letting him know that Gunde is on his way.

Gunde turns up to Jock's house at the expected time, helps Jock clean up the plant. He stays for a cup of tea and talks to Jock about his Olympic achievements.

After 2 days, the MATCHCare Server asks Jock to provide feedback on Gunde's service. Via his MATCHCare Requester App, Jock rates Gunde as on-time, efficient, polite and indicates that he would like Gunde to help him in the future. The MATCHCare Server records this information to make better connections in the future, in particular regarding Jock and Gunde.

The MATCHCare Server bills Jock's account and deposits money in Gunde's.

Scenario: Odd Jobs

Dave is going to dig a hole on Friday. It's going to be a big hole and he's going to need some help. He pulls out his OddJobs Requester App, built on the MATCH software infrastructure, and issues a request:

> Friday, 13:00–15:00, 3 people, "digging hole", semi-strenuous work, no specific skills or tools required.

Location information is automatically included.

Dave's request is sent to the OddJobMATCH Server, which searches through its database to find suitable candidates based on the location, time, the characteristics of the job, and ratings by previous requesters. The OddJobMATCH Server selects 10 suitable candidates and sends them invitations. The responders use their iPhone OddJobMATCH Responder Apps to either accept or decline the invitation. When 3 responders have accepted, the OddJobMATCH

Server stops sending out invitations and informs Dave of the details of the responders and lets the selected responders know when and where the job is. Pending invitations are cancelled.

After the job has been done, Dave uses his OddJobMATCH Requester App to inform the OddJobMATCH Server App that the job has been completed. Later, Dave is issued a bill via Klarna and credit is registered on the accounts of each of the workers. Finally, Dave is also invited to rate the quality of service of each of the workers, and they are can rate the job they had to do.

2 The MATCH Infrastructure

MATCH is a software infrastructure that provides mechanisms for matching users requiring information or services with potential suppliers of such information or services. The primary goal of the MATCH system is to connect people to other people in the most effective way. The core of the system acts as a *mediator* in establishing contacts: people can contact the system via a Smart Phone. Furthermore, based on the information about earlier established contacts and feedback from users, the system learns to bring people in contact with each other in the most effective way.

Two different client applications are generally required for any particular configuration of MATCH: one for requesters and one for responders. The *requester application* allows requesters to formulate new requests, to give feedback on responders, and to manage the requester's account. The *responder application* receives requests from the MATCH system and allows the responder to accept or decline these requests, optionally providing additional information such a time constraints. The precise form of the requester and responder applications depend on the specific configuration of the MATCH system.

MATCH Architecture

The software of MATCH can be conceptually divided into five parts: the *web front-end*, the *database*, the *contact engine*, *requester clients* and *responder clients*.

Web front-end acts as a configuration dashboard, via which typical domain data like users, groups, phone numbers, (e)mail addresses, services and scheduled jobs can be created, edited and deleted.[2]

Database stores the above data for each configuration of MATCH, along with user feedback and knowledge derived from contacts established earlier.[3]

[2]The web-front end does not need to be implemented. You will need need to come up with a way of populating your system in order to run it. Design in such a way that adding a web-front end should not require significant redesign.

[3]A database need not be designed. Your live objects can serve that role. Design in such a way that adding a proper database should not require significant redesign.

Contact Engine receives requests from requesters and finds appropriate responders to address those requests; manages all requester and responder information, feedback, preferences, etc.

Requester Client enables requesters to issue requests, monitor or cancel those requests, provide feedback on responders, and manage their MATCH account details.

Responder Client enables responders to receive invitations to respond to request, to provide feedback on requesters, if the configuration allows it, and to manage their MATCH account details.

The contact engine is the primary component of the system. It handles inbound and outbound communication with the system and provides the intelligent matching and scheduling functionality. The heart of the contact engine is the *request loop*. Requests loop through the system until they are fully completed, managed by the contact engine. Logically, the contact engine can be thought of as four components:[4]

Reception determines which steps must be taken by MATCH in order to fulfil (part of) a request.

Matcher searches for appropriate participants for a request.

Scheduler schedules requests based on job descriptions in the database.

These components will now be described in more detail.

Reception The major role of the Reception component is to determine what action should be taken by the MATCH system based on a request. A request could also originate from the Scheduler, for example, if the MATCH system contacts a user to ask for feedback or for availability as a MATCH responder for a certain time period. The Reception component is available for performing updates to the contents of the database in terms of adding previously unknown telephone numbers, adding feedback from users, or changing responder's schedules.

Matcher The Matcher component tries to find matching users for a request. For example, a person calling the MATCH system could ask for a connection with a specialist on a particular topic. Matching can be complicated, since the preferences and time schedules of the requester and the candidate responders must be taken into account, as well as feedback about earlier contacts.

[4]This describes one possible collection of components. Others are possible. Your system may start with these and diverge into another design.

Scheduler The Scheduler component realises the execution of various types of scheduled jobs. Typical jobs are:

- contacting potential responders about availability
- contacting requester about connection made
- contacting requesters and responders to obtain feedback about earlier connections

In executing these jobs, the Scheduler component keeps track of the time schedules and preferences of users. The Scheduler itself need not take part in the request loop: its messages could enter the request loop as if they come from outside the system. Jobs for the Scheduler can be put into the database manually via the web front-end, or automatically, as the result of the execution of requests in the contact engine.

3 Feature Requirements

This section presents various requirements of the different features of the system. Many of these descriptions are somewhat incomplete because the precise details of certain features depend on the specific application in which they are used.

These features are annotation to indicate which are most important for your design activity. Firstly, all features are compulsory, though the degree to which they are supported may vary, also depending upon what achievement you are aiming for. The following annotations are used:

R – only rudimentary support required.

B – must support end to end functionality for at least one configuration (Section 4).

X – must be elaborated in detail for higher-graded achievements.

Naturally, you may follow your interests and provide more than the minimum.

Requester (B)

The Requester App manages requests as they progress through various states.

Requests are created by filling in the different fields associated with a request (via appropriate user interface components). The simplest example could be a single button (Help!). Some local validation can occur. Requests may be canceled, in some configurations. Different request types may be available within a configuration, each with different data.

The Requester App allows past and pending requests to be managed.

The Requester App permits log in, account creation, account management, payment option. These are discussed elsewhere.

Feedback (optional) — given past completed request, provide feedback. Type of feedback is flexible, provided by some feedback description. Must be compatible with what MATCH expects.

Request Tracking (B)

Requests must be tracked through all their states, from initiation, to completion of the task, through receipt of feedback (if relevant) and closing of the request.

Request DB (R)

Requests should be stored in a database.

Responder (X)

The Responder App offers different features.

The most important is dealing with pending requests. Any request can be accepted or declined. In some cases, it may be possible to cancel an accepted request. In other cases, it may be possible to bid on a request (then either the MATCH system or the requester uses bids to select the best match).

The Responder App may also allow feedback to be given on any completed request.

The Responder App allows the users to log in (and log out), and to manage their accounts with the MATCH system, including viewing pending requests, payments due, and the request history.

Responder DB (R)

Responder information should be stored in a database.

Preferences (R)

Both requesters and responders may have preferences (soft constraints) that influence connections. Contrast preferences with hard constraints, such as timetable limitations: hard constraints must be satisfied, whereas preferences need not be satisfied, but should if they can be.

Preferences will be very configuration specific, but will cover things like time of activity, duration of activity, gender, age, topics considered, and so forth.

Preferences are handled within the matcher.

Matcher (X)

While the functionality of computing a single match is to be hidden behind the interface of the core match component, the system should offer considerable support for how the match component is used when handling a request. The match component should support matching a single responder or a bounded list of responders. Repeatedly calling the match component for a given request should result in different matches.

Different strategies can be used to find one or more matches:

- batch: process many requests together (e.g., on given date)
- match one at a time until there is success or no more matches
- match many and ask responders one by one until right number accepts
- match and invite many responders, cancel those who don't win/accept
- match and invite many, take bids, accept best
- as above, but client accepts possible match
- different strategies for retrying

When no (more) matches are available, a number of strategies are possible:

- panic/escalate — contact a human
- inform requester and cancel
- try again after some delay (number of retries, delay model (fixed, exponential increasing)).

The matcher tries to find several candidate responders and selects between them using one of several methods and based on several characteristics. Some of these are:

round robin — the Matcher 'randomly' selects a responder from the set of candidates available.

last spoken — the Matcher selects the responder that was selected previously.

rating — the Matcher uses feedback provided by the requester about potential responders and selects one with the highest rating.

friendly rating — the Matcher again selects based on the received ratings, but occasionally randomly selects a different responder in order to provide them with the opportunity to improve their rating.

geographically nearest — location of requester and responder is used to determine match

cheapest — cost is used

urgency — how quickly the request needs to be serviced

rating of human knowledge/skill — important in cases where the people request contact with specialists or service providers.

time schedule — indicate when certain people can be reached for certain purposes. The MATCH system can distinguish between regular planning and ad-hoc schedules caused by sudden events or delays.

combination — some weighted combination of these various factors.

Feedback (B)

Requester may give feedback on responders and vice versa, e.g., can say I like the shifts I was given, I don't like that responder, requester was difficult to work with/for, job was easy/too hard.

The details of the feedback can be configured for each specific MATCH configuration. A number of different categories of feedback can be offered and the answers can be presented on a number of different scales, all configurable.

The feedback is used mostly by MATCH to produce better matches, though it can be saved and used for auditing purposes.

Feedback may be disable for either or both parties.

On additional type of feedback can be provided. It should be possible to report abuse. The input to this feature could be simply a flag notifying of abuse or a text field where the details of the abuse is recorded. Any abuse reported must be immediately forwarded to the relevant humans managing the MATCH configuration, or to an external party (e.g., the police) handling such cases.

Learning (R)

The learning feature, which is part of MATCH, aims to determine good connections between requesters and responders *and* improve on them based on feedback. In general, learning from past experiences of all kinds and forecasting based on these experiences plays a crucial role in MATCH. To facilitate learning, all requesters and responders and their preferences need to be registered with the learner. In addition, all requests (active or otherwise) plus all feedback needs to pass to the learner.

For your project, the learning feature is a black box, not expected to be implemented (or even designed). However, it needs to be designed for and in mock implementations will need to do something sensible (e.g., randomly select match).

Login (R)

Both requesters and responders must log in to their respective apps before being connected to the server (authentication). All actions must be authorised, based on the user's identity and validity of a user's subscription.

Account management (R)

Both requesters and responders can manage their accounts, in particular managing their subscriptions, checking payments made and received, and so forth. Requesters and responders may be able to register to a corresponding MATCH system using the corresponding apps — though this feature may be disabled. It may be possible to create new accounts to join a MATCH system using the apps — again, this feature may be disabled.

User Management (R)

The MATCH system should offer facility for managing user accounts, including creating accounts, resetting passwords, disabling accounts, etc.

Payment Models (B)

The MATCH system should support a range of payment models. The standard model is that the requester pays and the responder is paid, and MATCH takes a percentage.

Different payment models are possible. These will be described for requester and responder separately.

Requester payment models:

- pay per use — requester pays each time a request is submitted
- pay per use + "entry fee" — requester also pays a fee to use the system
- subscription (per requester) — requester pays a monthly/yearly subscription to use the system
- subscription up to certain usage level, then pay per use — the subscription can be limited to a fixed number of uses (in total, per month, per week, ...); after these have been exhausted, the pay per use model takes over
- subscription up to certain usage level, then no further use — instead of shifting to a pay per use model, the system becomes inaccessible
- group subscription (for a set of users within some MATCH configuration — perhaps all members of a nursing home) — some users share a subscription; the rules of the subscription are applied collectively
- a company pays for a whole MATCH instance — requester use the system free of charge, and their company is billed, following the rules of the subscription
- free

Responder payment models:

- free — volunteer or already paid or payment model doesn't make sense for this configuration
- fixed rate per request
- hourly rate
- based on bidding/negotiation
- salary — fixed per time

Logging (B)

Due to the importance of (some) MATCH system applications (health impact, risk of abuse, risk of financial loss, etc), extensive logging of all activities is required to facilitate accurate audit.

In particular, all details of each step of each request need to be stored, including: the request id, parties involved in each step, timestamps (both on local app and in MATCH server), state of the request. Every interaction with the MATCH server (in- and out-going messages) must be logged. Decisions made by MATCH must be logged. Relevant internal messages between MATCH components must be logged, for messages that have a perceivable effect on the state of a request. The internal workings of the match do not need to be logged for auditing purposes, though it may be logged separately for debugging purposes.

Usage Statistics (R)

The MATCH server records various usage statistics for various users. Requester and Responder apps may also record local statistics.

Security (R)

All communications between client apps and the server must be secure. All user data must be protected. The log of events must be secure to avoid being tampered with.

4 MATCH Configurations

Following are a brief descriptions of a number of possible MATCH configurations. For each of these configurations, the way requesters make requests, the way invitations are sent to responders, the payment model, the feedback model, etc will be fixed. The number of users using each single MATCH configuration could vary from several hundreds to several thousands.

Job Hunting

In this configuration, the requesters are people requiring work, and the responders are companies providing work. The requests could include the kind of work being sort (part-time, permanent) and the qualifications of the applicant. The responder would similarly register job openings with the system giving details of the work (type of work, qualifications, number of hours/weeks vs. permanent). The MATCH configuration links requesters and responder jobs.

Workforce Planning

This configuration operates within the context of some company who uses MATCH to organise their work schedule. The requesters are employees wanting to work some shifts, and the responder is the company organising the work schedule. The employees register their preferences in terms of number of hours they want to work and when — employee skills etc are assumed to be already understood by the match system. The requester registers the number of slots to fill and the details of the work

A number of additional requirements govern the matching problems, such as upper and lower bounds on the amount employees can work each day/week and so forth.

Knowledge Sharing

In this configuration the requesters are people who seek expertise in some area, whether it be quantum physics or plumbing, and the responders are people who know stuff.

Social Care — Home-based Monitoring

This configuration connects people who are confined to their home and occasionally need help with small jobs, like cleaning up knocked-over pot plants, with regular people who willingly volunteer their time to help them.

In this situation, it is important to track requests that remain pending for too long.

Odd Jobs

This configuration connects people who need an odd job completed, no questions asked, with people lacking scruples. Security, in particular, anonymity and non-traceability, are important in this configuration.

Emergency Response

This configuration connects people in trouble with people who are trained to help them. It can be seen potentially as a supplement to 112.

Mail Distribution

This configuration connects individuals and companies with packages that need to be delivered somewhere to responders consisting of delivery companies. Various constraints are attached to requests, such as pick-up location, destination, delivery time constraints, size/weight of package. Responders typically want to batch various deliveries together, to get their trucks as full as possible, and reduce the amount of distance travelled, both when picking up packages and delivering packages – and in addition, they need to satisfy the delivery constraints made by requesters.

Financial Services

This configurations connects people in search of financial advice with people who provide financial advice. Different kinds of financial services may be available (stocks, bonds, options, etc) with different degrees of risk, and different financial institutions are available to provide such services.

Sports

This configuration handles requesters wanting to form a sports team (either long term or for a one-off event), to play some individual sport, or to find an emergency player for an existing team ("missing a goalie to play a football match starting in 15 min"). The requesters are connected with others wishing to play sport. Both individual and team sports are handled. Skill levels can also be specified. Some requests are specific (wanting to find an emergency player for a specific match), whereas others are open (willing to play at any time).

Note that the requester and responder roles are not necessarily clear cut in some configurations. For instance, someone may register that they want to play tennis at some time (making them more like a responder), whereas someone else may register that they want to play tennis today (making them more like a requester).

Dating

Both requesters and responders are people looking for a date (of various sorts). Thus requesters and responders are treated equally. Feedback is important.

Student and tutors

The system matches students needing tutor services. Tutoring could be answering a single question, more extensive coverage of the material in a single session, or Tutors are often also students, and have timetable constraints. The system must limit the access of individual students to tutors, to ensure fairness.

Study groups

This configuration matches students who want to form study groups either for an individual assignment or exam, or for the duration of a course. Feedback on other students can be specific ("I don't get on well with him") or general ("She knows her stuff").